Phase II CAQH CORE 270: Connectivity Rule
version 2.2.0 March 2011

Table of Contents

REVISION HISTORY FOR PHASE II CORE CONNECTIVITY RULE 4
1 BACKGROUND 5
1.1 Guiding Principles 5
2 ISSUES TO BE ADDRESSED AND BUSINESS JUSTIFICATION 5
2.1 CORE Phase II Connectivity Rule Background 6
2.2 CORE Phase II Connectivity - Decisions on Message Envelope Standards 6
2.2.1 Why Two Standards in Phase II? 7
2.2.2 Will CORE Move to One Standard in Future Phases? 7
2.3 Rationale for Basic Conformance Requirements 7
3 SCOPE 8
3.1 What the Rule Applies To 8
3.2 When the Rule Applies 10
3.3 When the Rule Does Not Apply 11
3.4 What the Rule Does Not Require 11
3.5 Technical Requirements and Assumptions 11
3.6 Outside the Scope of this Rule 12
3.7 Relationship to CORE Phase I Rule and Safe Harbor 12
4 RULE 13
4.1 Basic Conformance Requirements for Key Stakeholders 13
4.1.1 Health Plans and Health Plan Vendors 13
4.1.2 Clearinghouses, Health Information Exchanges and Other Intermediaries 13
4.1.3 Providers and Provider Vendors 14
4.1.4 Illustration of Conformance Requirements for Key Stakeholders 14
4.1.4.1 Envelope Standards 14
4.1.4.2 Submitter Authentication Standards 14
4.2 CORE-compliant Envelope Specifications using Message Enveloping Standards 15
4.2.1 Specifications for HTTP MIME Multipart (Envelope Standard A) 15
4.2.1.1 Real Time Request Message Structure (non-normative) 15
4.2.1.2 Real Time Response Message Structure (non-normative) 16
4.2.1.3 Batch Submission Message Structure (non-normative) 17
4.2.1.4 Batch Submission Response Message Structure (non-normative) 18
4.2.1.5 Batch Submission Acknowledgement Retrieval Request Message Structure (non-normative) 19
4.2.1.6 Batch Submission Acknowledgement Retrieval Response Message Structure (non-normative) 20
4.2.1.7 Batch Results Retrieval Request Message Structure (non-normative) 21
4.2.1.8 Batch Results Retrieval Response Message Structure (non-normative) 22
4.2.1.9 Batch Results Acknowledgement Submission Message Structure (non-normative) 23
4.2.1.10 Batch Results Acknowledgement Submission Response Message Structure (non-normative) 24
4.2.1.11 Envelope Processing Error Message Structure (non-normative) 25
4.2.1.12 Payload Attachment Handling 25
4.2.2 Specifications for SOAP+WSDL (normative) (Envelope Standard B) 25
4.2.2.1 CORE Phase II Connectivity XML Schema Specification (normative) 26
4.2.2.2 CORE Phase II Connectivity Web Services Definition Language (WSDL) Specification (normative) 29
4.2.2.3 Real Time Request Message Structure (non-normative) 32
4.2.2.4 Real Time Response Message Structure (non-normative) 33
4.2.2.5 Batch Submission Message (non-normative) 34
4.2.2.6 Batch Submission Response Message (non-normative) 35

4.2.2.7 Batch Submission Acknowledgement Retrieval Request Message (non-normative) *35*
4.2.2.8 Batch Submission Acknowledgement Retrieval Response Message (non-normative) *37*
4.2.2.9 Batch Results Retrieval Request Message (non-normative) *37*
4.2.2.10 Batch Results Retrieval Response Message (non-normative) *39*
4.2.2.11 Batch Results Acknowledgement Submission Message (non-normative) *39*
4.2.2.12 Batch Results Acknowledgement Submission Response Message (non-normative) *40*
4.2.2.13 ErrorMessage Structure (non-normative) *41*
4.2.2.14 Envelope Processing Error Message (non-normative) *41*
4.2.2.15 Payload Attachment Handling *42*
4.3 General Specifications Applicable to Both Envelope Methods ***42***
4.3.1 Request and Response Handling 42
4.3.1.1 Real Time Requests *42*
4.3.1.2 Batch Submission *42*
4.3.1.3 Batch Response Pickup *42*
4.3.2 Submitter Authentication and Authorization Handling 43
4.3.3 Error Handling 43
4.3.3.1 HTTP Status and Error Codes (Normative, Not Comprehensive) *44*
4.3.3.2 Envelope Processing Status and Error Codes (Normative, Comprehensive) *45*
4.3.3.3 Examples of Status and Error Codes (non-normative) *46*
4.3.3.4 Examples of Error Messages (non-normative) *46*
4.3.4 Audit Handling 46
4.3.4.1 Tracking of Date and Time and Payload ID *47*
4.3.5 Capacity Plan 47
4.3.5.1 Real Time Transactions *47*
4.3.5.2 Batch Transactions *47*
4.3.6 Response, Timeout and Retransmission Requirements 48
4.3.7 Publication of Entity-Specific Connectivity Guide 48
4.4 Envelope Metadata Fields, Descriptions, Intended Use and Syntax/Value-Sets ***49***
4.4.1 Message Envelope 49
4.4.2 Table of CORE Envelope Metadata 50
4.4.3 Enumeration of Processing Mode and PayloadType Fields 54
4.4.3.1 Real Time Transactions *54*
4.4.3.2 Batch Transactions *56*
4.4.4 Enumeration Convention for PayloadType when Handling Non-X12 Payloads (Non-normative) 61

5 CORE SAFE HARBOR **61**

6 APPENDIX **62**

6.1 Abbreviations and Definitions Used in this Rule ***62***
6.2 References ***68***
6.3 Sequence Diagrams ***69***
6.3.1 Real Time Interaction 69
6.3.2 Batch Interaction 70
6.3.2.1 Batch Interaction for Specific Payload Types *70*
6.3.2.2 Batch Interaction for Mixed Payload Types *72*
6.3.3 Generic Batch Retrieval Request and Receipt Confirmation 74
6.3.4 Generic Batch Submission with Batch Payload and Synchronous Payload Receipt Confirmation 75

REVISION HISTORY FOR PHASE II CORE CONNECTIVITY RULE

Version	Revision	Description	Date
2.0.0	Major	CORE Phase II Connectivity Rule, balloted and approved by CORE members.	July 15, 2008
2.0.1	Minor	Batch connectivity schemas and examples have been updated to eliminate gaps identified by early adopters.	Mar 16, 2009
Change Summary for Version 2.0.1 1. Section 4.2.1.3 • Removed "Request" from "Batch Submission Request Message Structure" specification title 2. Section 4.2.1.4 • Specification added 3. Section 4.2.1.5 - 4.2.1.6 • Original Batch Submission Acknowledgment Message Structure specification subdivided 4. Section 4.2.1.7 - 4.2.1.8 • "Results" added to specification titles 5. Section 4.2.1.9 - 4.2.1.10 • Specifications added 6. Section 4.2.2.5 • Removed "Request" from "Batch Submission Request Message Structure" specification title 7. Section 4.2.2.6 • Specification added 8. Section 4.2.2.7 - 4.2.2.8 • Original Batch Submission Acknowledgment Message Structure specification subdivided 9. Section 4.2.2.9 - 4.2.2.10 • "Results" added to specification titles 10. Section 4.2.1.11 - 4.2.1.12 • Specifications added 11. Section 6.3.1 • Real time interaction sequence diagram added 12. Section 6.3.2 • Batch interaction sequence diagram added			
2.2.0	Minor	1. Adjustments to support ASC X12 HIPAA-adopted v5010 Eligibility and Claim Status transactions to support PPACA Section 1104 2. Revisions to address Phase II FAQs 3. Updates to the SOAP+WSDL and HTTP+MIME examples to correct errors and omissions and to make the examples consistent with updated PayloadTypes table 4. Adjustments to optional and required field inconsistencies as described in the metadata table in the Connectivity Rule, as specified in the SOAP XSD schema and as presented in the examples for SOAP and/or HTTP+MIME under Schemas, Examples and Rule Text 5. MIME examples in Sections 4.2.1.1 through 4.2.1.9 were corrected to remove the filename attribute, which is not addressed by the CORE rule 6. Section 4.3.4.1 modified by removing the requirement for HTTP Header Date field, an incorrect holdover of text from previous rule version 7. MIME Examples in Sections 4.2.1.3 through 4.2.1.10 and SOAP Examples in Sections 4.2.2.5 through 4.2.2.12 updated to show MIME and SOAP examples using only ASC X12 270/271 payload type values rather than mixed batch payload type values 8. Section 6.3.2 expanded to show a new sequence diagram for specific payload types in addition to the Mixed Payload Type sequence interaction diagram that already exist	Mar 18, 2011

1 BACKGROUND

This rule addresses the message envelope metadata, the message envelope standards and the submitter authentication standards for both batch and real time transactions, and communications-level errors and acknowledgements. This rule is designed to provide a "safe harbor" that the application vendors, providers and health plans (or other information sources) can be assured will be supported by any CORE-certified trading partner. All CORE-certified organizations must demonstrate the ability to implement connectivity as described in this rule. This rule is not intended to require trading partners to remove existing connections that do not match the rule, nor is it intended to require that all CORE trading partners must use this method for all new connections. CORE expects that in some technical circumstances, trading partners may agree to use different communication mechanism(s) and/or security requirements than that described by this rule.

1.1 Guiding Principles

The following CORE guiding principles apply to the CORE Phase II Connectivity Rule:

- CORE will not create or promote proprietary approaches to electronic interactions/transactions.
- CORE will suggest migration steps to promote successful and timely adoption of CORE rules.
- To promote interoperability, rules will be built upon HIPAA, and CORE will coordinate with other key industry bodies (for example, ASC X12 and the Blue Cross and Blue Shield Association).
- Where appropriate, CORE will address the emerging interest in XML, or other evolving standards.
- Whenever possible, CORE has used existing market research and proven rules. CORE rules reflect lessons learned from other organizations that have addressed similar issues.
- CORE rules will support the Guiding Principles of HHS's National Health Information Network (NHIN).
- CORE will not build a switch, database, or central repository of information.
- All CORE recommendations and rules will be vendor neutral.
- Rules will not be based on the least common denominator but rather will encourage feasible Phase II progress.
- CORE will promote and encourage voluntary adoption of the rules.
- CORE participants do not support "phishing."
- CORE rules address both *Batch* and *Real time* (these terms are defined in *Appendix 6.1: Abbreviations and Definitions used in this Rule*), with a movement towards Real time (Hence, Phase II certification related to batch does not apply to entities that do not do batch transactions.)
- All of the Phase II rules are expected to evolve in future phases.
- CORE's Connectivity rules are written such that they can accommodate not only Eligibility transactions, but also can apply to any other administrative transaction.
- Acknowledging connectivity support for Eligibility, Claim Status and other administrative transactions in light of the Patient Protection and Affordable Care Act (ACA) §1104.

2 ISSUES TO BE ADDRESSED AND BUSINESS JUSTIFICATION

Currently, multiple connectivity methods – some based on open standards, others on proprietary approaches – are in use for administrative electronic transactions in the healthcare industry. Healthcare providers and health plans support multiple connectivity methods to connect to different health plans, clearinghouses, provider organizations and others. Supporting multiple connectivity methods for administrative electronic transactions adds costs for health plans and providers. Connectivity and Security standards from standards development organizations such as OASIS, W3C, and IETF are intended to be industry neutral, allowing for implementation variations, and thus are not specific enough to provide interoperability in healthcare. For example, several open standards for enveloping, such as SOAP and ebMS exist

in the marketplace. These open standards are industry neutral and hence do not define the metadata, vocabularies and semantics needed to support industry specific transactions. Further complicating this issue is the wide variance in implementations of these standards in the healthcare market, the continuing use of proprietary approaches and the industry's developing use of the public internet.

The Committee on Operating Rules for Information Exchange (CORE®) Connectivity & Security Subgroup aims to fill this gap by adopting existing open standards and formulating Connectivity/Security Rules that provide industry specific (i.e., for healthcare administrative electronic transactions) guidance on using these open standards. The CORE Connectivity Rule was developed using a consensus-based approach among industry stakeholders, and is designed to facilitate interoperability, improve utilization of administrative transactions, enhance efficiency and lower the cost of information exchange in healthcare.

The following sections describe the work completed while developing the CORE Phase I Connectivity Rule, and the outstanding issues that are being addressed by the CORE Phase II Connectivity Rule. The rationale for the selected envelope and authentication standards within the CORE Phase II Connectivity Rule is also presented.

2.1 CORE Phase II Connectivity Rule Background

CORE Phase I (i.e., *CORE Operating Rule 153: Connectivity Rule*) defined a Connectivity/Security Rule, which is a safe harbor that required the use of the HTTP/S transport protocol over the public Internet. It also specified a minimum set of metadata outside the ASC X12 payload (e.g., date/time, payload ID, and other elements), and aspects of connectivity/security such as response times, acknowledgements and errors. Since the CORE Phase I Connectivity Rule is a safe harbor, CORE Phase I-certified entities are required to support the adopted CORE Phase I Connectivity method at a minimum, but may also implement additional other connectivity/security methods. A set of Conformance Tests were defined for CORE Phase I Connectivity certification and many organizations have attained CORE Phase I certification.

The CORE Phase I Connectivity Rule (i.e., *CORE Operating Rule 153: Connectivity Rule*) required the use of HTTP/S over the public Internet. CORE made a decision to limit the rule at this high-level to provide a first step toward Connectivity, understanding that later phases of CORE would issue more detailed requirements. CORE was aware the Phase I Rule did not provide the optimum level of specificity for implementations as it was developed as a first step. CORE Phase I Connectivity-certified implementations were based on many types of enveloping methods: HTTP POST with name/value pairs, HTTP MIME Multipart, W3C XML Schema and SOAP+WSDL among others. Further, within each of these envelope method implementations, significant variations exist in field names and locations of Phase I Connectivity metadata, message envelope structure, authentication methods, routing approaches and security related information. Variations among enveloping methods and metadata pose a major challenge for interoperability.

The CORE Phase II Connectivity charge was to create a definitive Rule that would further facilitate interoperability. Such interoperability is expected to improve efficiencies and utilization of electronic transactions, and ultimately lower the administrative costs for healthcare providers and payers.

2.2 CORE Phase II Connectivity - Decisions on Message Envelope Standards

The CORE Phase II Connectivity efforts were focused on creating a definitive safe harbor that reached the envelope level. As part of CORE Phase II, the Connectivity & Security Subgroup performed extensive analysis of open standards that are available for enveloping the payload (ASC X12 or other types of data). The available open standards were rated against the agreed upon CORE Phase II Connectivity criteria and this rating process was used to select the envelope standards that met the large majority of these criteria; first creating a short-list and then reducing that short-list to two envelope standards that met the overwhelming majority of the criteria:

A. HTTP MIME Multipart

B. SOAP + WSDL

Over several months, discussions were held on the relative merits of these two envelope standards to determine if there is a clear winner among the two. SOAP+WSDL supports interface definition with a XSD schema/WSDL, automated development/validation, and is well aligned with standards adoption within healthcare industry bodies such as HL7 and HITSP. Furthermore, the SOAP+WSDL methodology lends itself to future Rule development using Web-Services standards for more advanced requirements like reliability. HTTP MIME Multipart on the other hand provides a relatively

simple and well understood protocol framework and a lower performance overhead relative to SOAP, and has a large implementation base within this industry; including many of the CORE Phase I certified entities.

Having analyzed the advantages and challenges of the two envelope standards, CORE then analyzed several CORE Phase I Connectivity-compliant real world implementations of the above two envelope standards. The real world examples confirmed that both the envelope standards are in widespread use in this industry, and both perform well under real-world transaction volumes. Hence, the real-world implementation analysis did not point to a clear winner among the two envelope standards.

2.2.1 Why Two Standards in Phase II?

After extensive analysis, the two envelope standards (HTTP MIME Multipart and SOAP+WSDL) selected by the CORE Phase II Connectivity & Security Subgroup from the initial long list of standards were shown to:

- meet the CORE Phase II Connectivity criteria;
- have significant installed base in this industry; and,
- perform well under real world transaction loads.

Since both these standards have significant merits, the Subgroup debated the advantages and challenges of having a single envelope standard versus both these envelope standards as part of the CORE Phase II Rule and Safe Harbor. The major advantage of a Rule based on a single envelope standard is that it would be more definitive and facilitate better interoperability. However, having just one standard would require implementers of the other envelope standard (i.e., the one that was not chosen) to modify their implementations to be CORE Phase II-compliant. Since both standards met the criteria and have large installed bases, convergence on a single standard would create a barrier to adoption of CORE Phase II Connectivity Rule by a large segment of the industry. Moreover, the two standards have many similarities that the market is still exploring.

Based on the above analysis, the consensus view of the Subgroup was that a CORE Phase II Connectivity Rule should be based on both envelope standards. This provides CORE with a path to facilitate adoption in a market that is still maturing, and where many still require education. CORE recognizes that two standards do not support interoperability as much as one, however a CORE Phase II Rule and Safe Harbor based on two envelope standards would facilitate adoption while also improving interoperability relative to the current state of the industry. Given the current state of the industry, where the number of variations in the envelope standards and metadata is extremely large, reducing the number of envelope methods to two is a significant step forward in facilitating interoperability, improving efficiencies, improving utilization and hence lowering administrative cost for healthcare providers and health plans. Additionally, CORE is offering a path to use two standard market implementations – an option that does not exist today.

2.2.2 Will CORE Move to One Standard in Future Phases?

In the interest of further facilitating interoperability, CORE expects to move towards a single envelope standard in future phases. Given the current state of healthcare connectivity (i.e., use of many distinct connectivity methods), creating a CORE Phase II Connectivity rule with two envelope methods vastly improves the state of the market, while also providing an opportunity for education and greater experience with two standards that meet the growing market needs for connectivity. Taking this phased step enables the healthcare industry to make a more informed decision as it considers supporting a single envelope, safe harbor standard in future CORE phases.

The choice of two envelope standards provides this industry an opportunity to monitor the marketplace for movement towards a specific standard. Holding industry discussions on Phase II implementations will be essential to ensure that CORE continues to be aligned with the direction of related national healthcare initiatives.

2.3 Rationale for Basic Conformance Requirements

Supporting two envelope standards and two submitter authentication methods as part of the CORE Phase II Connectivity Rule makes it necessary to specify the conformance requirements for stakeholders. The rationale for these basic conformance requirements is based on the following guiding principles and assumptions.

It is assumed that the typical message exchange patterns are:

- *Real time*: Health plan receives a Real time request directly (or relayed via a Clearinghouse) from Providers and responds synchronously (as part of the same connection). A sequence diagram of the Real time Interaction is provided in Appendix 6.3.
- *Batch*: Health plan receives a Batch submission directly (or relayed via a Clearinghouse) from a Provider. Provider polls Health plan (directly or indirectly via a Clearinghouse) at a later time and receives an acknowledgement for the batch submission. Provider polls Health plan (directly or indirectly via a Clearinghouse) at a later time and retrieves the batch results as response. Provider then sends an acknowledgement to Health plan (directly or indirectly via a Clearinghouse) that the results were received with/without errors. A sequence diagram of the Batch Interaction is provided in Appendix 6.3.

In both of these message exchange patterns, Provider acts as a client, and Health Plan acts as a server, and Clearinghouse acts as both client and server.

One of the goals of CORE Connectivity is to improve utilization of electronic transactions by enabling more entities to interoperate with other entities, including reducing the implementation barrier for small entities (e.g., small providers).

Based on the above guiding principle and assumption about typical message exchange patterns, the rationale for the conformance requirements is as follows:

- Envelope standards: Organizations that receive and process or relay the requests (i.e., as a server) are required to support both envelope methods to facilitate connectivity from multiple clients. Generally organizations implementing a server have higher technical capabilities, and the difference in complexity between HTTP MIME Multipart and SOAP+WSDL are usually not significant for such organizations.
- Submitter authentication standards: Organizations that receive and process (or relay) requests (i.e., as a server) generally enforce a specific authentication method to control access to their resources. Supporting this authentication method is a credential issuance and management scheme defined by an organizational policy. The complexity of supporting two such policies and credential management mechanisms is high at the entity where submitter authentication is enforced (server), but is relatively low at the submitter (client). For this reason, server-side implementations are only required to support one of the two submitter authentication methods. To connect to the server, the client implementation needs to be able to authenticate itself to the server using the authentication method that is enforced at the server.

3 SCOPE

3.1 What the Rule Applies To

The technical scope of Phase II CORE Connectivity Rule can be described in terms of the specific network layers within the Open Systems Interconnection Basic Reference Model[1] (OSI model). As shown in the diagram below, the scope of Phase II CORE Connectivity Rule is OSI Layers 3 and 4 (Transport and Network layers) and OSI Layers 5 and 6 (Session and Presentation layers, also called Message Encapsulation layers). The Phase I CORE Connectivity Rule (i.e., *CORE Operating Rule 153: Connectivity Rule*) defined a Safe Harbor in terms of OSI Layers 3 and 4 (Transport and Network layers). CORE Phase II Connectivity Rule builds on the Phase I foundation and provides a more definitive Rule for encapsulating the metadata that is required for routing, identification/authentication and auditing.

[1] Zimmerman, H., OSI Reference Model – ISO Model of Architecture for Open Systems Interconnection, IEEE Transactions on Communications, Vol. Com-28, No. 4, April 1980.

Figure #3.3.1

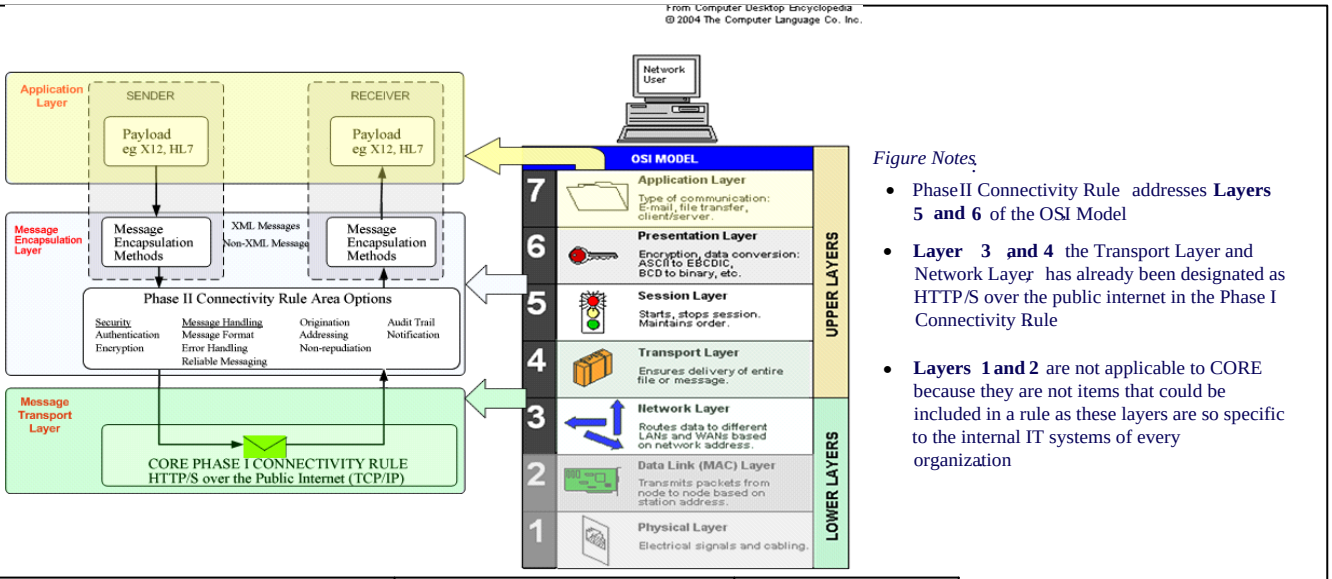


Figure Notes

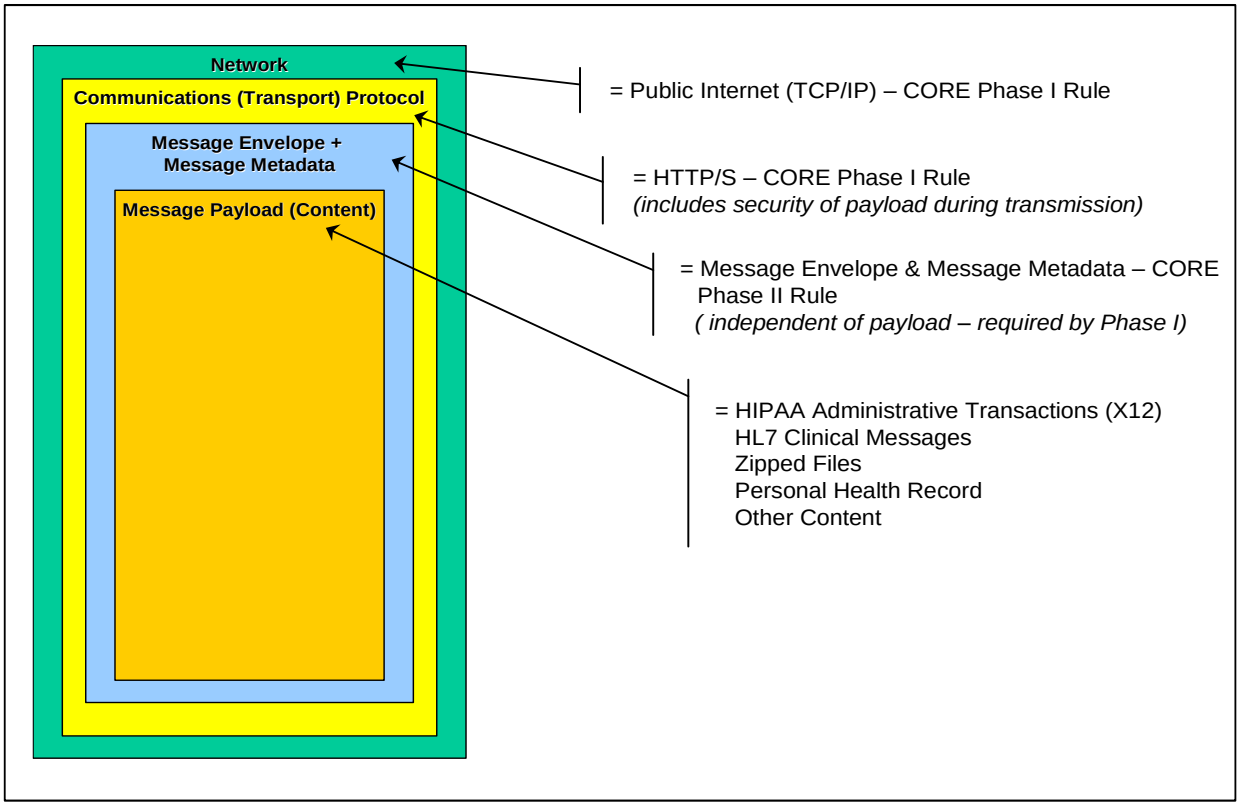
- Phase II Connectivity Rule addresses **Layers 5 and 6** of the OSI Model
- **Layer 3 and 4** the Transport Layer and Network Layer has already been designated as HTTP/S over the public internet in the Phase I Connectivity Rule
- **Layers 1 and 2** are not applicable to CORE because they are not items that could be included in a rule as these layers are so specific to the internal IT systems of every organization

OSI Model		Messaging Infrastructure Model	Addressed in
Application Layer	OSI7	Application Layer	Payload Standards (e.g., X12, HL7)
Presentation Layer	OSI6	Message Encapsulation Layer (envelope)	Phase II
Session Layer	OSI5		Phase II
Transport Layer	OSI4	Message Transport Layer	Phase I
Network Layer	OSI3		Phase I

As shown in the Figure 3.3.1 above, typically an application file (or Payload) such as X12 or HL7 is created or processed by an application that resides in the Application Layer (Layer 7 in the OSI Model). The Message Encapsulation layer (Layers 5 and 6 in the OSI Model) create a message envelope, and handle connectivity and security. The underlying layers (Layers 1 through 4) provide the necessary message transport and the network infrastructure (e.g., TCP/IP is provided at Layer 3).

As shown in Figure #3.3.2 below, the Message Envelope is outside the Message Payload (content), and inside the Transport Protocol envelope. Here, the Transport Protocol Envelope corresponds to OSI Model Layer 3 and 4, Message Envelope corresponds to OSI Model Layers 5 and 6, and Message Payload (content) corresponds to OSI Model Layer 7. The Phase I CORE 153 Connectivity Rule version 1.1.0 was based on the use of HTTP/S as the transport protocol over the public Internet; hence the transport protocol envelope consists of HTTP headers. Examples of message payload include HIPAA administrative transactions (ASC X12), HL7 clinical messages, zipped files, etc.

Figure #3.3.2



The following is a list of standards and their versions that this Rule is based on:

- HTTP Version 1.1
- SSL Version 3.0
  - This does not preclude the optional use of TLS 1.0 (or a higher version as required for FIPS 140 compliance) for connectivity with trading partners that require FIPS 140 compliance. CORE Connectivity certification requires testing with SSL 3.0 for transport security.
- MIME Version 1.0
- The MIME Multipart/Form-Data (IETF RFC 2388)
- SOAP Version 1.2
- WSDL Version 1.1
- Web Services-Security 1.1

3.2 When the Rule Applies

The Phase II CORE Connectivity Rule is applicable to Eligibility Benefit Inquiry and Response (270/271), Health Care Claim Status Request and Response (276/277), and Health Care Claim Acknowledgement (277) payloads in Real time and Batch mode, and may also be applied to other payload types. Note, some entities may also apply the rule to other ASC X12 administrative transactions. Phase II CORE Connectivity Rule is a Safe Harbor, and therefore only needs to be used if mutually agreed to by the trading partners. It is expected that in some instances, other or existing mechanisms may be more appropriate methods of Connectivity.

3.3 When the Rule Does Not Apply

The Phase II CORE Connectivity Rule **DOES NOT** apply in the following scenarios:

- When CORE-certified entities exchange payloads other than Eligibility, Claim Status and Claim Status-related payloads. This rule is designed to be payload agnostic, and as such it is expected that CORE-certified entities will use this methodology for payloads other than Eligibility, Claim Status and Claim Status-related payloads; however, the rule does not require this.
- When trading partners mutually agree to use a non-CORE connectivity mechanism.

3.4 What the Rule Does Not Require

The Phase II CORE Connectivity Rule:

- **DOES NOT** require trading partners to discontinue existing connections that do not match the rule.
- **DOES NOT** require that trading partners must use a CORE-complaint method for all new connections.
- **DOES NOT** require that all CORE trading partners use only one method for all connections.
- **DOES NOT** require any CORE-certified entity to do business with any trading partner or other CORE-certified entity.

Further, the Phase II CORE Connectivity Rule **DOES NOT** require the following:

- Additional centralized services other than those that are already provided in the Internet (e.g., Domain name and TCP/IP routing services).
- Additional directories or data repositories.
- Additional centralized Public Key Infrastructure (PKI) Certificate Authorities, identity management or authentication servers.
- Use of specific hardware platforms, software or programming languages.

3.5 Technical Requirements and Assumptions

The following technical requirement applies to this rule:

- The use of the public Internet for HTTP/S transport as specified in *Phase I CORE 153 Connectivity Rule.*

The following assumptions apply to this rule:

- Interoperability, utilization and efficiency will improve by having fewer connectivity/security variations and uniform enveloping standards and metadata.
- The typical message exchange patterns for Real time and Batch transactions are as described in §2.3.
- Health Plans and Clearinghouses generally have greater technical and infrastructure capabilities than most Providers.
- The difference in complexity between HTTP MIME Multipart and SOAP+WSDL is not significant for a server implementation.
- The Server-side enforcement of submitter authentication is far more complex than supporting the submitter authentication method on the client-side.
- This Rule is based upon a specific set of open standards and the versions of these standards specified in §3.1. As open standards and versions evolve, appropriate version control practices may need to be applied to keep the Rule consistent with industry best practices with regards to standard versions.
- This rule is a component of the larger set of Phase II CORE Rules; as such, all the CORE Guiding Principles apply to this rule and all other rules.

- All entities seeking Phase II certification will be Phase I certified, or concurrently testing for compliance with Phase I rules, as Phase I provides a foundation for Phase II CORE. The exception is vendors/clearinghouses that do not conduct the v5010 270/271 eligibility, or the 276/277 claim status, or the 277 claim acknowledgement transactions.

3.6 Outside the Scope of this Rule

The following items are outside the scope of this rule:

- The use of the message envelope and metadata defined in this rule for those messages that are sent over TCP/IP connections that are private (e.g., Intranet, leased lines, or VPN).
- Non-TCP/IP protocols such as packet switching (e.g., X.25, SNA, and Frame Relay).
- Application of this rule to other administrative transactions, such as 837 claims, as well as clinical or other transactions, e.g., HITSP interoperability specifications.
- Submitter Authorization is a local decision at the site that receives a request.
- The list of trusted Certificate Authorities is a decision between trading partners.
- The maximum size of a batch file that is accepted by a Server. The Server implementer may publish its file size limit, if any, in its Connectivity Companion Guide. (See §4.3.7)

3.7 Relationship to CORE Phase I Rule[2] and Safe Harbor

This Phase II Connectivity Rule extends the Phase I Connectivity Rule (i.e., *CORE Operating Rule 153: Connectivity Rule*) and establishes a Safe Harbor by further specifying the connectivity that all CORE-certified organizations must demonstrate and implement. See §5 CORE Safe Harbor. Each of the Phase I requirements has been incorporated in the Phase II Rule except that the HTTP error code for security authorization failures has been superseded under the SOAP option under this Rule. Under the Phase II SOAP option, security authorization failures will follow the SOAP standard.

Since the Phase II CORE Connectivity Safe Harbor and Rule is more definitive than the CORE Phase I Safe Harbor Connectivity Rule, those entities that achieve CORE Phase II Connectivity compliance are assumed to be CORE Phase I compliant, but such entities will not be required to support CORE Phase I connections. Should entities ask for CORE Phase I solutions rather than Phase II solutions, trading partners will mutually determine which option will be used.

Phase I Connectivity Rule elements re-used in this Phase II Rule:

- Transport standard (HTTP/S) (Phase I *CORE Operating Rule 153 Connectivity Rule*)
- Acknowledgements (Phase I *CORE Operating Rule 150 Batch Acknowledgements, CORE Operating Rule 151 Real Time Acknowledgements*)
- Response Time, Time Out, Re-transmission (Phase I *CORE Operating Rule 155 Batch Response Time Rule, CORE Operating Rule 156 Real Time Response Rule*)

Phase I Connectivity Rule (i.e., Phase I *CORE Operating Rule 153 Connectivity Rule*) elements modified in this Phase II Rule:

- Date/time syntax (to make it UTC standards compliant)

Phase I Connectivity Rule (i.e., Phase I *CORE Operating Rule 153 Connectivity Rule*) elements extended in this Phase II Rule:

- Envelope structure, metadata names and syntax
- Attachment handling

[2] Phase I CORE 153 Connectivity Rule.

- Authentication (e.g., *UserName* token for SOAP envelope)

4 RULE

This section specifies the basic conformance requirements, the envelope metadata and the specifications for HTTP MIME Multipart and SOAP+WSDL. The rationale and business justification for these conformance requirements are described in §2.3.

4.1 Basic Conformance Requirements for Key Stakeholders

Like the Phase I Connectivity Rule (CORE Operating Rule 153: Connectivity Rule), the Phase II Connectivity Rule is a Safe Harbor. CORE-certified entities are required to comply with all CORE Rules and thus a CORE-certified entity must support a CORE-compliant connectivity method. However, as a Safe Harbor the CORE Connectivity Rule:

- **DOES NOT** require trading partners to discontinue existing connections that do not match the rule.
- **DOES NOT** require that trading partners must use a CORE-complaint method for all new connections.
- **DOES NOT** require that all CORE trading partners use only one method for all connections.
- **DOES NOT** require any CORE-certified entity to do business with any trading partner or other CORE-certified entity.

The terms used in this Section, such as Client, Server, Envelope Standard A and B, Submitter Authentication Standard C and D, are defined in §6.1 *Abbreviations and Definitions used in this Rule*. The sections below specify the conformance requirements for stakeholders that can be CORE-certified, including Health Plans, Clearinghouses, Health Information Exchanges, and other intermediaries, Providers, Provider Vendors and Health Plan Vendors.

4.1.1 Health Plans and Health Plan Vendors

Health Plans or Health Plan Vendors (Servers) must implement capability to support (See §6.1) both Message Envelope Standards (A and B). For each envelope standard, the following are the conformance requirements for Real time and Batch transactions:

- Real time[3]: Required for ASC X12 v5010 270/271 and ASC X12 v5010 276/277 transactions
- Batch: Optional for ASC X12 v5010 270/271 and ASC X12 v5010 276/277 transactions; must be supported if Batch is offered for ASC X12 v5010 270/271 and ASC X12 v5010 276/277 transactions

Health Plans or Health Plan Vendors (Server) must implement and enforce one of the two (C or D) Submitter Authentication Standards for Phase II CORE Connectivity Compliance (for Real time and/or Batch transactions). If a Health Plan or Health Plan Vendor implements a client (e.g., for plan-to-plan messaging), then for such clients, the Health Plan or Health Plan Vendor must implement the capability to support one of the two Message Envelope Standards (A or B), and must implement support for both Submitter Authentication Standards (C and D).

4.1.2 Clearinghouses, Health Information Exchanges and Other Intermediaries

Intermediaries, including Clearinghouses, Switches, and Health Information Exchanges, act as both Client and Server. The Server portion of Clearinghouses/Switches/Health Information Exchanges must implement the capability to support both Message Envelope Standards (A and B). The Client portion of Clearinghouses/Switches/Health Information Exchanges must implement the capability to support one of the two Message Envelope Standards (A or B). For each envelope standard (A and B), the following are the conformance requirements for Real time and Batch transactions:

- Real time: Required for ASC X12 v5010 270/271 and ASC X12 v5010 276/277 transactions
- Batch: Optional for ASC X12 v5010 270/271 and ASC X12 v5010 276/277 transactions; must be supported if Batch is offered for ASC X12 v5010 270/271 and ASC X12 v5010 276/277 transactions

[3] Real time and Batch are defined in §6.1 Abbreviations and Definitions used in this Rule.

The Client portion of Clearinghouses/Switches/Health Information Exchanges must implement both (C and D) Submitter Authentication Standards for CORE Phase II Connectivity Compliance (for Real time and/or Batch transactions) to connect to Health Plans (i.e., as a client of the Health Plan). The Server portion of Clearinghouses/Switches/Health Information Exchanges must implement one of the two Authentication Standards for authenticating the Providers that submit requests to them (See Figure #4.1.6.1 for illustration).

If there is more than one intermediary (e.g., Clearinghouse/Switch/Health Information Exchange) between a Provider and Health Plan, then the Server portion of each intermediary must implement both Message Envelope Standards (A and B), and one of the two Submitter Authentication Standards (C or D). Further, the Client portion of each intermediary must implement one of the two Message Envelope Standards (A or B), and the Client portion must implement both the Submitter Authentication Standards (C and D).

4.1.3 Providers and Provider Vendors

Providers or Provider Vendors (Clients) must implement one of the two (A or B) Envelope Standards for Phase II CORE Connectivity Compliance. If a Provider or Provider Vendor implements a Server, then it must support both envelope methods (A and B) on the Server. For the Envelope Standard (A or B) implemented, the following are the conformance requirements for Real time and Batch transactions:

- Real time: Required for ASC X12 v5010 270/271 and ASC X12 2 v5010 76/277 transactions
- Batch: Optional for ASC X12 v5010 270/271 and ASC X12 v5010 276/277 transactions; must be supported if Batch is offered for ASC X12 v5010 270/271 and ASC X12 v5010 276/277 transactions

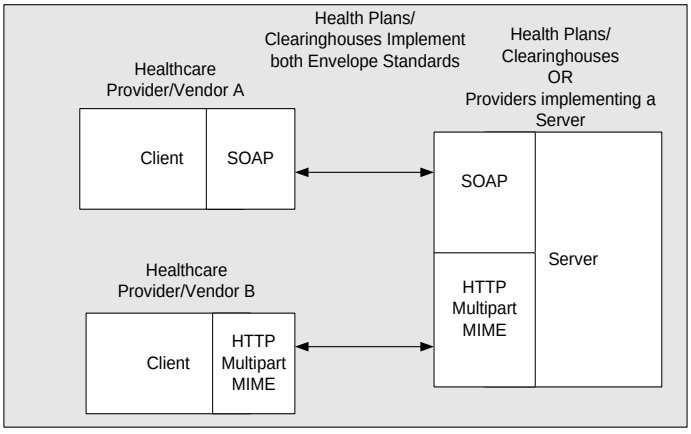
Providers or Provider Vendors must implement both (C and D) Submitter Authentication Standards for Phase II CORE Connectivity Compliance (for Real time and/or Batch transactions). If a Provider or Provider Vendor implements a Server, then it must support one of the two Submitter Authentication Standards on the Server.

4.1.4 Illustration of Conformance Requirements for Key Stakeholders

4.1.4.1 Envelope Standards

Figure #4.1.6.1 below shows the Phase II CORE Connectivity Rule's envelope standards conformance requirements for key stakeholders. Health Plans and Clearinghouses/Switches/Information Exchanges that conform to Phase II CORE Connectivity Rule must implement both envelope standards (SOAP+WSDL and HTTP MIME Multipart). Healthcare Providers or Provider Vendors must implement one of the envelope standards.

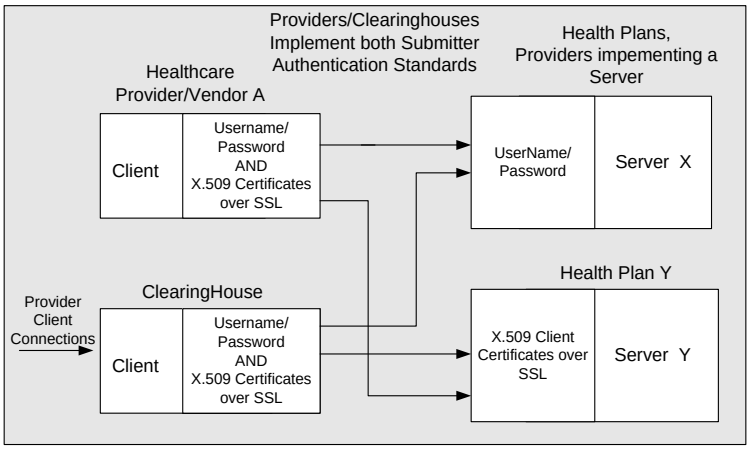
Figure #4.1.6.1



4.1.4.2 Submitter Authentication Standards

Figure #4.1.6.2 below shows the submitter authentication conformance requirements for key stakeholders. As shown, the Health Plans (servers) implement one of the two submitter Authentication Standards. Healthcare Providers/Provider Vendors and Clearinghouse components that handle submissions to Health Plans must implement both submitter Authentication Standards (i.e., only the client portion of authentication).

Figure #4.1.6.2



4.2 CORE-compliant Envelope Specifications using Message Enveloping Standards

CORE is not creating its own envelope standards, but rather, supports the use of existing standards. Note: The terms *normative* and *non-normative* are defined in *§6.1 Abbreviations and Definitions used in this Rule.*

4.2.1 Specifications for HTTP MIME Multipart (Envelope Standard A)

Unlike in the SOAP+WSDL (i.e., Envelope B) case, the HTTP MIME[4] Multipart envelope does not provide a standard Schema specification that is normative (definitive) and can be verified in an automated manner. For this reason, HTTP MIME Multipart Real time Request/Response examples below are non-normative[5]. They are based on the real-world examples provided by CORE participants, and have been updated to use the CORE-compliant metadata.

4.2.1.1 Real Time Request Message Structure (non-normative)

The following is an example of a Real time request message using the HTTP MIME Multipart envelope method. The HTTP Header is shown in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
POST /core/eligibility HTTP/1.1
Host: server_host:server_port
Content-Length: 2408
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_270_Request_005010X279A1
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

RealTime
--XbCY
Content-Disposition: form-data; name="PayloadID"

e51d4fae-7dec-11d0-a765-00a0c91e6da6
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="UserName"
```

[4] MIME Multipart is defined in IETF RFC 2388 (http://www.faqs.org/rfcs/rfc2388.html)
[5] The lack of a normative schema specification means that client-server interfaces need to be manually created and validated (i.e., automated interface generation and validation is not supported). This is not considered a significant limitation since the request/response interfaces are relatively simple to implement and validate.


```
hospa
--XbCY
Content-Disposition: form-data; name="Password"

8y6dt3dd2
--XbCY
Content-Disposition: form-data; name="SenderID"

HospitalA
--XbCY
Content-Disposition: form-data; name="ReceiverID"

PayerB
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY
Content-Disposition: form-data; name="Payload"

<contents of file go here -- 1674 bytes long as specified above>
--XbCY--
```

4.2.1.2 Real Time Response Message Structure (non-normative)

The following is an example of a Real time response message using the HTTP MIME Multipart envelope method. The portion of the request below that is colored in blue is the HTTP Header. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
HTTP/1.1 200 OK
Content-Length: 2408
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_271_Response_005010X279A1
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

RealTime
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0c91e6da6
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="SenderID"

PayerB
--XbCY
Content-Disposition: form-data; name="ReceiverID"

HospitalA
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY
Content-Disposition: form-data; name="ErrorCode"

Success
--XbCY
Content-Disposition: form-data; name="ErrorMessage"

None
--XbCY
Content-Disposition: form-data; name="Payload"

<contents of file go here -- 1674 bytes long as specified above>
--XbCY--
```


4.2.1.3 Batch Submission Message Structure (non-normative) [6]

The following is an example of a Batch Submission message using the HTTP MIME Multipart envelope method. The HTTP Headers are shown colored in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
POST /core/eligibility HTTP/1.1
Host: server_host:server_port
Content-Length: 244508
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_270_Request_005010X279A1
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

Batch
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0d91e6fa6
--XbCY
Content-Disposition: form-data; name="PayloadLength"

10240
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="UserName"

hospa
--XbCY
Content-Disposition: form-data; name="Password"

8y6dt3dd2
--XbCY
Content-Disposition: form-data; name="SenderID"

HospitalA
--XbCY
Content-Disposition: form-data; name="ReceiverID"

PayerB
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY
Content-Disposition: form-data; name="Checksum"

6A3FE55946
--XbCY
Content-Disposition: form-data; name="Payload"

<contents of batch file go here>
--XbCY--
```

[6] Generic Batch Submission Request (§6.3.4) uses the same request message as the Batch Submission Request message structure above, with *PayloadType* values based on what is being submitted.

4.2.1.4 Batch Submission Response Message Structure (non-normative)[7]

The following is an example of a synchronous response to a Batch Submission request message using the HTTP MIME Multipart envelope method. The HTTP Headers are shown colored in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
HTTP/1.1 200 OK
Content-Length: 2408
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_BatchReceiptConfirmation
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

Batch
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0c91e6da6
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="SenderID"

PayerB
--XbCY
Content-Disposition: form-data; name="ReceiverID"

HospitalA
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY
Content-Disposition: form-data; name="ErrorCode"

Success
--XbCY
Content-Disposition: form-data; name=”ErrorMessage”

None
--XbCY--
```

[7] Generic Batch Submission Response (§6.3.4) uses the same response message as the Batch Submission Response message structure depicted below, with *PayloadType* values based on the response to what is being submitted.

4.2.1.5 Batch Submission Acknowledgement Retrieval Request Message Structure (non-normative)

The following is an example of a Batch Submission Acknowledgement Retrieval request message using the HTTP MIME Multipart envelope method. The HTTP Headers are shown colored in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
POST /core/eligibility HTTP/1.1
Host: server_host:server_port
Content-Length: 244508
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_999_RetrievalRequest_005010X231A1
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

Batch
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0d91e6fa6
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="UserName"

hospa
--XbCY
Content-Disposition: form-data; name="Password"

8y6dt3dd2
--XbCY
Content-Disposition: form-data; name="SenderID"

HospitalA
--XbCY
Content-Disposition: form-data; name="ReceiverID"

PayerB
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY--
```


4.2.1.6 Batch Submission Acknowledgement Retrieval Response Message Structure (non-normative)

The following is an example of a Batch Submission Acknowledgement Retrieval Response message using the HTTP MIME Multipart envelope method. The HTTP Headers are shown colored in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
HTTP/1.1 200 OK
Content-Length: 12648
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_999_Response_005010X231A1[8]
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

Batch
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0c91e6da6
--XbCY
Content-Disposition: form-data; name="PayloadLength"

10240
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="SenderID"

PayerB
--XbCY
Content-Disposition: form-data; name="ReceiverID"

HospitalA
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY
Content-Disposition: form-data; name="Checksum"

6A3FE55946
--XbCY
Content-Disposition: form-data; name="Payload"

<contents of batch file go here>
--XbCY
Content-Disposition: form-data; name="ErrorCode"

Success
--XbCY
Content-Disposition: form-data; name="ErrorMessage"

None
--XbCY--
```

[8] Although this example shows a 999 payload type being sent as a response from a server to the client, this could also include a TA1. Alternatively, the server may elect to send only a TA1 without any functional group.

4.2.1.7 Batch Results Retrieval Request Message Structure (non-normative)[9]

The following is an example of a Batch Results Retrieval request message using the HTTP MIME Multipart envelope method. The HTTP Headers are shown colored in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
POST /core/eligibility HTTP/1.1
Host: server_host:server_port
Content-Length: 244508
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_005010_Request_Batch_Results_271
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

Batch
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0d91e6fa6
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="UserName"

hospa
--XbCY
Content-Disposition: form-data; name="Password"

8y6dt3dd2
--XbCY
Content-Disposition: form-data; name="SenderID"

HospitalA
--XbCY
Content-Disposition: form-data; name="ReceiverID"

PayerB
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY--
```

[9] Generic Batch Retrieval Request (§6.3.3) uses the same request message as the Batch Results Retrieval Request message structure depicted below, with different *PayloadType* values based on what is being retrieved.

4.2.1.8 Batch Results Retrieval Response Message Structure (non-normative) [10]

The following is an example of a Batch Retrieval Response message using the HTTP MIME Multipart envelope method. The HTTP Headers are shown colored in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
HTTP/1.1 200 OK
Content-Length: 12648
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_271_Response_005010X279A1
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

Batch
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0c91e6da6
--XbCY
Content-Disposition: form-data; name="PayloadLength"

10240
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="SenderID"

PayerB
--XbCY
Content-Disposition: form-data; name="ReceiverID"

HospitalA
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
None
--XbCY
Content-Disposition: form-data; name="Checksum"

6A3FE55946
--XbCY
Content-Disposition: form-data; name="Payload"
<contents of batch file go here>
--XbCY
Content-Disposition: form-data; name="ErrorCode"

Success
--XbCY
Content-Disposition: form-data; name="ErrorMessage"

None
--XbCY--
```

[10] Generic Batch Retrieval Response (§6.3.3) uses the same response message as the Batch Results Retrieval Response message structure depicted below, with different *PayloadType* values based on the response to what is being retrieved.

4.2.1.9 Batch Results Acknowledgement Submission Message Structure (non-normative)[11]

The following is an example of a Batch Results Acknowledgement Submission message using the HTTP MIME Multipart envelope method. The HTTP Headers are shown colored in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
POST /core/eligibility HTTP/1.1
Host: server_host:server_port
Content-Length: 244508
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_999_SubmissionRequest_005010X231A1[12]
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

Batch
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0d91e6fa6
--XbCY
Content-Disposition: form-data; name="PayloadLength"

10240
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="UserName"

hospa
--XbCY
Content-Disposition: form-data; name="Password"

8y6dt3dd2
--XbCY
Content-Disposition: form-data; name="SenderID"

HospitalA
--XbCY
Content-Disposition: form-data; name="ReceiverID"

PayerB
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY
Content-Disposition: form-data; name="Checksum"

6A3FE55946
--XbCY
Content-Disposition: form-data; name="Payload"

<contents of batch file go here>
--XbCY--
```

[11] Generic Batch Receipt Confirmation (§6.3.3) uses the same request message as the Batch Results Acknowledgement Submission message structure depicted below, with different *PayloadType* values as appropriate.

[12] Although this example shows a 999 payload type being submitted by the client to the server, this could also include a TA1. Alternatively, the client may elect to submit only a TA1 without any functional group.

4.2.1.10 Batch Results Acknowledgement Submission Response Message Structure (non-normative) [13]

The following is an example of a Batch Results Acknowledgement Submission Response message using the HTTP MIME Multipart envelope method. The HTTP Headers are shown colored in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
HTTP/1.1 200 OK
Content-Length: 2408
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

X12_Response_ConfirmReceiptReceived
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

Batch
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0d91e6fa6
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="SenderID"

PayerB
--XbCY
Content-Disposition: form-data; name="ReceiverID"

HospitalA
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY
Content-Disposition: form-data; name="ErrorCode"

Success
--XbCY
Content-Disposition: form-data; name=”ErrorMessage”

None
--XbCY--
```

[13] Generic Batch Receipt Confirmation Response (§6.3.3) uses the same request message as the Batch Results Acknowledgement Submission Response message structure depicted below, with different *PayloadType* values as appropriate.

4.2.1.11 Envelope Processing Error Message Structure (non-normative)

The following is an example of an Envelope Processing Error message using the HTTP MIME Multipart envelope method. The HTTP Headers are shown colored in blue. The remainder of the request (shaded in light gray) is the body of the MIME Multipart message.

```
HTTP/1.1 200 OK
Content-Length: 2408
Content-Type: multipart/form-data; boundary=XbCY

--XbCY
Content-Disposition: form-data; name="PayloadType"

COREEnvelopeError
--XbCY
Content-Disposition: form-data; name="ProcessingMode"

RealTime
--XbCY
Content-Disposition: form-data; name="PayloadID"

f81d4fae-7dec-11d0-a765-00a0a91e6fa6
--XbCY
Content-Disposition: form-data; name="TimeStamp"

2007-08-30T10:20:34Z
--XbCY
Content-Disposition: form-data; name="SenderID"

PayerB
--XbCY
Content-Disposition: form-data; name="ReceiverID"

HospitalA
--XbCY
Content-Disposition: form-data; name="CORERuleVersion"

2.2.0
--XbCY
Content-Disposition: form-data; name="ErrorCode"

VersionMismatch
--XbCY
Content-Disposition: form-data; name="ErrorMessage"

Expected Version X, received version Y
--XbCY—
```

4.2.1.12 Payload Attachment Handling

The Payload must be sent as a MIME Multipart attachment for both Real time as well as Batch transactions.

4.2.2 Specifications for SOAP+WSDL (normative[14]) (Envelope Standard B)

This section defines the SOAP+WSDL envelope method for CORE Phase II Connectivity. The XML Schema that is defined below is used within the Web Services Definition Language (WSDL) specification.

Note: The terms SOAP, WSDL, MTOM, Normative and Non-normative are defined in *Appendix §6.1: Abbreviations and Definitions used in this Rule*.

[14] See §6.1 Abbreviations and Definitions used in this Rule for a definition of Normative.

4.2.2.1 CORE Phase II Connectivity XML Schema Specification (normative)

The CORE Phase II compliant XML Schema Specification file name below is called *CORERule2.2.0.xsd*, and is available at https://www.caqh.org/sites/default/files/core/wsdl/CORERule2.2.0.xsd. This schema has ten elements, each representing a type of request or response message envelope:

- Real time Request Schema (Element name is *COREEnvelopeRealTimeRequest*)
- Real time Response (Element name is *COREEnvelopeRealTimeResponse*)
- Batch Submission (Element name is *COREEnvelopeBatchSubmission*)
- Batch Submission Response (Element name is *COREEnvelopeBatchSubmissionResponse*)
- Batch Submission Acknowledgement Retrieval Request (Element name is *COREEnvelopeBatchSubmissionAckRetrievalRequest*)
- Batch Submission Acknowledgement Retrieval Response (Element name is *COREEnvelopeBatchSubmissionAckRetrievalResponse*)
- Batch Results Retrieval Request (Element name is *COREEnvelopeBatchResultsRetrievalRequest*)
- Batch Results Retrieval Response (Element name is *COREEnvelopeBatchResultsRetrievalResponse*)
- Batch Results Acknowledgement Submission (Element name is *COREEnvelopeBatchResultsAckSubmission*)
- Batch Results Acknowledgement Submission Response (Element name is *COREEnvelopeBatchResultsAckSubmission Response*)


```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd"
targetNamespace="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
  <xs:element name="COREEnvelopeRealTimeRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ProcessingMode" type="RealTimeMode" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="Payload" type="xs:string" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="COREEnvelopeRealTimeResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ProcessingMode" type="RealTimeMode" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="Payload" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="ErrorCode" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ErrorMessage" type="xs:string" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="COREEnvelopeBatchSubmission">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ProcessingMode" type="BatchMode" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadLength" type="xs:int" minOccurs="1" maxOccurs="1"/>
        <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="CheckSum" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="Payload" type="xs:base64Binary" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="COREEnvelopeBatchSubmissionResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ProcessingMode" type="BatchMode" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadLength" type="xs:int" minOccurs="0" maxOccurs="1"/>
        <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="CheckSum" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Payload" type="xs:base64Binary" minOccurs="0" maxOccurs="1"/>
        <xs:element name="ErrorCode" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ErrorMessage" type="xs:string" minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="COREEnvelopeBatchSubmissionAckRetrievalRequest">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ProcessingMode" type="BatchMode" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadLength" type="xs:int" minOccurs="0" maxOccurs="1"/>
```



```
      <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CheckSum" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="Payload" type="xs:base64Binary" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="COREEnvelopeBatchSubmissionAckRetrievalResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ProcessingMode" type="BatchMode" minOccurs="1" maxOccurs="1"/>
      <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="PayloadLength" type="xs:int" minOccurs="0" maxOccurs="1"/>
      <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CheckSum" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="Payload" type="xs:base64Binary" minOccurs="0" maxOccurs="1"/>
      <xs:element name="ErrorCode" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ErrorMessage" type="xs:string" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="COREEnvelopeBatchResultsRetrievalRequest">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ProcessingMode" type="BatchMode" minOccurs="1" maxOccurs="1"/>
      <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="PayloadLength" type="xs:int" minOccurs="0" maxOccurs="1"/>
      <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CheckSum" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="Payload" type="xs:base64Binary" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="COREEnvelopeBatchResultsRetrievalResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ProcessingMode" type="BatchMode" minOccurs="1" maxOccurs="1"/>
      <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="PayloadLength" type="xs:int" minOccurs="0" maxOccurs="1"/>
      <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CheckSum" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="Payload" type="xs:base64Binary" minOccurs="0" maxOccurs="1"/>
      <xs:element name="ErrorCode" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ErrorMessage" type="xs:string" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="COREEnvelopeBatchResultsAckSubmission">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ProcessingMode" type="BatchMode" minOccurs="1" maxOccurs="1"/>
      <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="PayloadLength" type="xs:int" minOccurs="0" maxOccurs="1"/>
      <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
      <xs:element name="CheckSum" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="Payload" type="xs:base64Binary" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
```



```
    </xs:complexType>
  </xs:element>
  <xs:element name="COREEnvelopeBatchResultsAckSubmissionResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PayloadType" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ProcessingMode" type="BatchMode" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="PayloadLength" type="xs:int" minOccurs="0" maxOccurs="1"/>
        <xs:element name="TimeStamp" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="SenderID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="ReceiverID" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="CORERuleVersion" type="xs:string" minOccurs="1" maxOccurs="1"/>
        <xs:element name="CheckSum" type="xs:string" minOccurs="0" maxOccurs="1"/>
        <xs:element name="Payload" type="xs:base64Binary" minOccurs="0" maxOccurs="1"/>
        <xs:element name="ErrorCode" type="xs:string"  minOccurs="1" maxOccurs="1"/>
        <xs:element name="ErrorMessage" type="xs:string"  minOccurs="1" maxOccurs="1"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:simpleType name="RealTimeMode">
    <xs:restriction base="xs:string">
      <xs:pattern value="RealTime"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="BatchMode">
    <xs:restriction base="xs:string">
      <xs:pattern value="Batch"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

4.2.2.2 CORE Phase II Connectivity Web Services Definition Language (WSDL) Specification (normative)

The CORE Phase II Connectivity Web Services Definition Language (WSDL) file below is called *CORERule2.2.0.wsdl*, and is available at *www.caqh.org/SOAP/WSDL/CORERule2.2.0.wsdl*. The WSDL below makes use of the XML Schema (CORERule2.2.0.xsd) as specified in §4.2.2.1. Within this WSDL the following types of messages are defined:

- Real time Request Message (Message name is *RealTimeRequestMessage*)
- Real time Response Message (Message name is *RealTimeResponseMessage*)
- Batch Submission Request Message (Message name is *BatchSubmissionMessage*)
- Batch Submission Response Message (Message name is *BatchSubmissionResponseMessage*)
- Batch Submission Acknowledgement Retrieval Request (Message name is *BatchSubmissionAckRetrievalRequestMessage*)
- Batch Submission Acknowledgement Retrieval Response (Message name is *BatchSubmissionAckRetrievalResponseMessage*)
- Batch Results Retrieval Request Message (Message name is *BatchResultsRetrievalRequestMessage*)
- Batch Results Retrieval Response Message (Message name is *BatchResultsRetrievalResponseMessage*)
- Batch Results Acknowledgement Submission Message (Message name is *BatchResultsAckSubmissionMessage*)
- Batch Results Acknowledgement Submission Response Message (Message name is *BatchResultsAckSubmissionResponseMessage*)

Using the above message definitions, the following types of transactions are defined:

- Real time Transaction (Operation name is *RealTimeTransaction*)
- Batch Submit Transaction (Operation name is *BatchSubmitTransaction*)
- Batch Submit Acknowledgement Retrieval Transaction (Operation name is *BatchSubmitAckRetrievalTransaction*)
- Batch Results Retrieval Transaction (Operation name is *BatchResultsRetrievalTransaction*)

- Batch Results Acknowledgement Transaction (Operation name is *BatchResultsAckSubmitTransaction*)
- Generic Batch Retrieval Transaction (Operation name is *GenericBatchRetrievalTransaction)*
- Generic Batch Receipt Confirmation Transaction (Operation name is *GenericBatchReceiptConfirmationTransaction)*
- Generic Batch Submission Transaction (Operation name is *GenericBatchSubmissionTransaction*)

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:CORE="http://www.caqh.org/SOAP/WSDL/"
                              xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
                              xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                              xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                              xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                              xmlns:CORE-XSD="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd"
                              xmlns="http://schemas.xmlsoap.org/wsdl/"
                              name="CORE"
                              targetNamespace="http://www.caqh.org/SOAP/WSDL/">
  <wsdl:types>
    <xsd:schema xmlns="http://schemas.xmlsoap.org/wsdl/"
                                          elementFormDefault="qualified"
                                          targetNamespace="http://www.caqh.org/SOAP/WSDL/">
      <xsd:import namespace="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd"
schemaLocation="CORERule2.2.0.xsd"/>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="RealTimeRequestMessage">
    <wsdl:part name="body" element="CORE-XSD:COREEnvelopeRealTimeRequest"/>
  </wsdl:message>
  <wsdl:message name="RealTimeResponseMessage">
    <wsdl:part name="body" element="CORE-XSD:COREEnvelopeRealTimeResponse"/>
  </wsdl:message>
  <wsdl:message name="BatchSubmissionMessage">
    <wsdl:part name="body" element="CORE-XSD:COREEnvelopeBatchSubmission"/>
  </wsdl:message>
  <wsdl:message name="BatchSubmissionResponseMessage">
    <wsdl:part name="body" element="CORE-XSD:COREEnvelopeBatchSubmissionResponse"/>
  </wsdl:message>
  <wsdl:message name="BatchSubmissionAckRetrievalRequestMessage">
    <wsdl:part name="body"
                element="CORE-XSD:COREEnvelopeBatchSubmissionAckRetrievalRequest"/>
  </wsdl:message>
  <wsdl:message name="BatchSubmissionAckRetrievalResponseMessage">
    <wsdl:part name="body"
              element="CORE-XSD:COREEnvelopeBatchSubmissionAckRetrievalResponse"/>
  </wsdl:message>
  <wsdl:message name="BatchResultsRetrievalRequestMessage">
    <wsdl:part name="body"
                  element="CORE-XSD:COREEnvelopeBatchResultsRetrievalRequest"/>
  </wsdl:message>
  <wsdl:message name="BatchResultsRetrievalResponseMessage">
    <wsdl:part name="body"
                  element="CORE-XSD:COREEnvelopeBatchResultsRetrievalResponse"/>
  </wsdl:message>
  <wsdl:message name="BatchResultsAckSubmissionMessage">
    <wsdl:part name="body" element="CORE-XSD:COREEnvelopeBatchResultsAckSubmission"/>
  </wsdl:message>
  <wsdl:message name="BatchResultsAckSubmissionResponseMessage">
    <wsdl:part name="body" element="CORE-XSD:COREEnvelopeBatchResultsAckSubmissionResponse"/>
 </wsdl:message>
  <wsdl:portType name="CORETransactions">
    <wsdl:operation name="RealTimeTransaction">
      <wsdl:input message="CORE:RealTimeRequestMessage"/>
      <wsdl:output message="CORE:RealTimeResponseMessage"/>
    </wsdl:operation>
    <wsdl:operation name="BatchSubmitTransaction">
      <wsdl:input message="CORE:BatchSubmissionMessage"/>
      <wsdl:output message="CORE:BatchSubmissionResponseMessage"/>
    </wsdl:operation>
   <wsdl:operation name="GenericBatchSubmissionTransaction">
      <wsdl:input message="CORE:BatchSubmissionMessage"/>
      <wsdl:output message="CORE:BatchSubmissionResponseMessage"/>
    </wsdl:operation>
```



```
    <wsdl:operation name="BatchSubmitAckRetrievalTransaction">
      <wsdl:input message="CORE:BatchSubmissionAckRetrievalRequestMessage"/>
      <wsdl:output message="CORE:BatchSubmissionAckRetrievalResponseMessage"/>
    </wsdl:operation>
    <wsdl:operation name="BatchResultsRetrievalTransaction">
      <wsdl:input message="CORE:BatchResultsRetrievalRequestMessage"/>
      <wsdl:output message="CORE:BatchResultsRetrievalResponseMessage"/>
    </wsdl:operation>
    <wsdl:operation name="BatchResultsAckSubmitTransaction">
      <wsdl:input message="CORE:BatchResultsAckSubmissionMessage"/>
      <wsdl:output message="CORE:BatchResultsAckSubmissionResponseMessage"/>
    </wsdl:operation>
<wsdl:operation name="GenericBatchRetrievalTransaction">
      <wsdl:input message="CORE:BatchResultsRetrievalRequestMessage"/>
      <wsdl:output message="CORE:BatchResultsRetrievalResponseMessage"/>
    </wsdl:operation>
    <wsdl:operation name="GenericBatchReceiptConfirmationTransaction">
      <wsdl:input message="CORE:BatchResultsAckSubmissionMessage"/>
      <wsdl:output message="CORE:BatchResultsAckSubmissionResponseMessage"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="CoreSoapBinding" type="CORE:CORETransactions">
    <soap12:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="RealTimeTransaction">
      <soap12:operation soapAction="RealTimeTransaction" style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="BatchSubmitTransaction">
      <soap12:operation soapAction="BatchSubmitTransaction" style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
<wsdl:operation name="GenericBatchSubmissionTransaction">
      <soap12:operation
                          soapAction="GenericBatchSubmissionTransaction" style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="BatchSubmitAckRetrievalTransaction">
      <soap12:operation
                          soapAction="BatchSubmitAckRetrievalTransaction" style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="BatchResultsRetrievalTransaction">
      <soap12:operation
                          soapAction="BatchResultsRetrievalTransaction" style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="BatchResultsAckSubmitTransaction">
      <soap12:operation
                            soapAction="BatchResultsAckSubmitTransaction" style="document"/>
      <wsdl:input>
```



```
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="GenericBatchRetrievalTransaction">
      <soap12:operation
                          soapAction="GenericBatchRetrievalTransaction" style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
     <wsdl:operation name="GenericBatchReceiptConfirmationTransaction">
      <soap12:operation
                            soapAction="GenericBatchReceiptConfirmationTransaction"
style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="Core">
    <wsdl:port name="CoreSoapPort" binding="CORE:CoreSoapBinding">
      <soap12:address location="http://URL_OF_WEB_SERVICE"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

The following sections show Request and Response messages using the SOAP envelope, based on the WSDL schemas defined above. The SOAP Real time Request/Response examples below are non-normative[15]. They are based on the real-world examples provided by CORE participants, but have been updated to use the CORE-recommended metadata that is part of CORE Phase II Connectivity.

4.2.2.3 Real Time Request Message Structure (non-normative)

The Real time Request message structure shown below specifies SOAP 1.2.

SOAP version 1.2 must be implemented by all Servers.

When sending or receiving payloads which contain non-printable characters, e.g., separator characters in an ASC X12 Interchange payload or in a non-ASC X12 Interchange payload in Real time using SOAP, the payload must be Base64 encoded.

This shows the following components:

1. The HTTP Headers are shown colored in blue.
2. The WS-Security Username and Password token (shown here with a pink background) is added to the SOAP Header by the platform on which SOAP is run. The SOAP platform's Web-Services Security Extensions may be configured to insert these tokens.
3. The portion of the SOAP envelope colored in green has the remaining metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)

[15] A non-normative description is informational only. See §6.1 Abbreviations and Definitions Used in this Rule.


```
POST /core/eligibility HTTP/1.1
Host: server_host:server_port
Content-Type: application/soap+xml; charset=UTF-8; action="RealTimeTransaction"

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
       <soapenv:Header>
       <wsse:Security
       xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-
       1.0.xsd" soapenv:mustUnderstand="true">
       <wsse:UsernameToken xmlns:wsu=http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
       wssecurity-utility-1.0.xsd wsu:Id="UsernameToken-21621663">
       <wsse:Username>bob</wsse:Username>
       <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-
       token-profile-1.0#PasswordText">bobPW</wsse:Password>
       </wsse:UsernameToken>
       </wsse:Security>
       </soapenv:Header>
        <soapenv:Body>
              <ns1:COREEnvelopeRealTimeRequest
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                <PayloadType> X12_270_Request_005010X279A1</PayloadType>
                      <ProcessingMode>RealTime</ProcessingMode>
                      <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                      <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                      <SenderID>HospitalA</SenderID>
                      <ReceiverID>PayerB</ReceiverID>
                      <CORERuleVersion>2.2.0</CORERuleVersion>
         <Payload><![CDATA[ISA*00* *00* *ZZ*NEHEN780 *ZZ*NEHEN003 ...IEA*1*000000031]]></Payload>
              </ns1:COREEnvelopeRealTimeRequest>
                </soapenv:Body>
</soapenv:Envelope>
```

4.2.2.4 Real Time Response Message Structure (non-normative)

The Real time Response message structure shown below specifies SOAP 1.2. The HTTP Header is shown in blue. The remainder of the request is the SOAP Envelope. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml;
action="http://www.caqh.org/SOAP/WSDL/CORETransactions/RealTimeTransactionResponse";charset=UTF-8

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
       <soapenv:Body>
              <ns1:COREEnvelopeRealTimeResponse
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0xsd">
                      <PayloadType>X12_271_Response_005010X279A1</PayloadType>
                      <ProcessingMode>RealTime</ProcessingMode>
                      <PayloadID>a81d44ae-7dec-11d0-a765-00a0c91e6ba0</PayloadID>
                      <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                      <SenderID>PayerB</SenderID>
                      <ReceiverID>HospitalA</ReceiverID>
                      <CORERuleVersion>2.2.0</CORERuleVersion>
           <Payload><![CDATA[ISA*00* *00* *ZZ*NEHEN780 *ZZ*NEHEN003
...IEA*1*000000031]]></Payload>
                      <ErrorCode>Success</ErrorCode>
                      <ErrorMessage></ErrorMessage>
                </ns1:COREEnvelopeRealTimeResponse>
         </soapenv:Body>
  </soapenv:Envelope>
```


4.2.2.5 Batch Submission Message (non-normative)[16]

The Batch Submission message structure shown below specifies SOAP 1.2, and also uses MTOM (See §6.1) to send the payload file. This shows the following:

1. The HTTP Headers are shown colored in blue.
2. The WS-Security Username and Password token (shown here with a pink background) is added to the SOAP Header by the platform on which SOAP is run. The SOAP platform's Web-Services Security Extensions may be configured to insert these tokens.
3. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)
4. The Batch file (MTOM attachment) is shown colored in orange.

```
POST /core/eligibilityBatch HTTP/1.1
Content-Type: multipart/related; boundary=MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614;
type="application/xop+xml"; start="0.urn:uuid:5117AAE1116EA8B87A1200060184615@apache.org"; start-
info="application/soap+xml"; action="BatchSubmitTransaction"

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:5117AAE1116EA8B87A1200060184615@apache.org>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
       <soapenv:Header>
       <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
       secext-1.0.xsd" soapenv:mustUnderstand="true">
       <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
       wssecurity-utility-1.0.xsd" wsu:Id="UsernameToken-21621664">
       <wsse:Username>bob</wsse:Username>
       <wsse:Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
       profile-1.0#PasswordText">bobPW</wsse:Password>
       </wsse:UsernameToken>
       </wsse:Security>
       </soapenv:Header>
       <soapenv:Body>
              <ns1:COREEnvelopeBatchSubmission
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                     <PayloadType>X12_270_Request_005010X279A1</PayloadType>
                     <ProcessingMode>Batch</ProcessingMode>
                     <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                     <PayloadLength>1551254</PayloadLength>
                     <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                     <SenderID>HospitalA</SenderID>
                     <ReceiverID>PayerB</ReceiverID>
                     <CORERuleVersion>2.2.0</CORERuleVersion>
                     <CheckSum>43B8485AB5</CheckSum>
                   <Payload>
                   <xop:Include       href="cid:1.urn:uuid:5117AAE1116EA8B87A1200060184692@apache.org"
                   xmlns:xop="http://www.w3.org/2004/08/xop/include" />
                     </Payload>
              </ns1:COREEnvelopeBatchSubmission>
         </soapenv:Body>
  </soapenv:Envelope>

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <1.urn:uuid:5117AAE1116EA8B87A1200060184692@apache.org>

<Mixed batch file>
```

[16] The Generic Batch Submission Request (§6.3.4) uses the same request message as the Batch Submission Request message structure depicted below, with *PayloadType* values based on what is being submitted.


```
--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```

4.2.2.6 Batch Submission Response Message (non-normative)[17]

The Batch Submission Response message structure shown below specifies SOAP 1.2 and MTOM. This shows the following:

1. The HTTP Headers are shown colored in blue.
2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: multipart/related; boundary=MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339;
type="application/xop+xml"; start="0.urn:uuid:0B72121B1FEFA9BDD31200060195340@apache.org"; start-
info="application/soap+xml";
action="http://www.caqh.org/SOAP/WSDL/CORETransactions/BatchSubmitTransactionResponse"

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:0B72121B1FEFA9BDD31200060195340@apache.org>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Body>
              <ns1:COREEnvelopeBatchSubmissionResponse
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                    <PayloadType>X12_BatchReceiptConfirmation</PayloadType>
                    <ProcessingMode>Batch</ProcessingMode>
                    <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                    <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                    <SenderID>PayerB</SenderID>
                    <ReceiverID>HospitalA</ReceiverID>
                    <CORERuleVersion>2.2.0</CORERuleVersion>
                    <ErrorCode>Success</ErrorCode>
                    <ErrorMessage></ErrorMessage>
                </ns1:COREEnvelopeBatchSubmissionResponse>
         </soapenv:Body>
  </soapenv:Envelope>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339--
```

4.2.2.7 Batch Submission Acknowledgement Retrieval Request Message (non-normative)

The Batch Submission Acknowledgement Retrieval Request message structure shown below specifies SOAP 1.2. The use of MTOM in Batch mode request/response creates multipart MIME even though there is no payload. This shows the following:

1. The HTTP Headers are shown colored in blue.
2. The WS-Security Username and Password token (shown here with a pink background) is added to the SOAP Header by the platform on which SOAP is run. The SOAP platform's Web-Services Security Extensions may be configured to insert these tokens.
3. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)

[17] The Generic Batch Submission Response (§6.3.4) uses the same response message as the Batch Submission Response message structure depicted below, with *PayloadType* values based on the response to what is being submitted.


```
POST /core/eligibilityBatch HTTP/1.1
Content-Type:      multipart/related;      boundary=MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614;
type="application/xop+xml";     start="0.urn:uuid:5117AAE1116EA8B87A1200060184615@apache.org";     start-
info="application/soap+xml"; action="BatchSubmitAckRetrievalTransaction"

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:5117AAE1116EA8B87A1200060184615@apache.org>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
       <soapenv:Header>
       <wsse:Security     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
       secext-1.0.xsd"                                                    soapenv:mustUnderstand="true">
       <wsse:UsernameToken           xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
       wssecurity-utility-1.0.xsd"                                      wsu:Id="UsernameToken-21621664">
       <wsse:Username>bob</wsse:Username>
       <wsse:Password       Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
       profile-1.0#PasswordText">bobPW</wsse:Password>
       </wsse:UsernameToken>
       </wsse:Security>
       </soapenv:Header>
       <soapenv:Body>
              <ns1:COREEnvelopeBatchSubmissionAckRetrievalRequest
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                     <PayloadType>X12_999_RetrievalRequest_005010X231A1</PayloadType>
                     <ProcessingMode>Batch</ProcessingMode>
                     <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                     <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                     <SenderID>HospitalA</SenderID>
                     <ReceiverID>PayerB</ReceiverID>
                     <CORERuleVersion>2.2.0</CORERuleVersion>
              </ns1:COREEnvelopeBatchSubmissionAckRetrievalRequest>
        </soapenv:Body>
  </soapenv:Envelope>
--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```

4.2.2.8 Batch Submission Acknowledgement Retrieval Response Message (non-normative)[18]

The Batch Submission Acknowledgement Retrieval Response message structure shown below specifies SOAP 1.2 and MTOM. This shows the following:

1. The HTTP Headers are shown colored in blue.
2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)
3. The MTOM Attachment is colored in orange.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type:       multipart/related;       boundary=MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339;
type="application/xop+xml";      start="0.urn:uuid:0B72121B1FEFA9BDD31200060195340@apache.org";      start-
info="application/soap+xml"; action="BatchSubmitAckRetrievalTransaction"

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:0B72121B1FEFA9BDD31200060195340@apache.org>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Body>
              <ns1:COREEnvelopeBatchSubmissionAckRetrievalResponse
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                    <PayloadType>X12_999_Response_005010X231A1</PayloadType>
                    <ProcessingMode>Batch</ProcessingMode>
                    <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                    <PayloadLength>1551254</PayloadLength>
                    <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                    <SenderID>PayerB</SenderID>
                    <ReceiverID>HospitalA</ReceiverID>
                    <CORERuleVersion>2.2.0</CORERuleVersion>
                    <CheckSum>43B8485AB5</CheckSum>
                   <Payload>
                  <xop:Include         href="cid:1.urn:uuid:5117AAE1116EA8B87A1200060184692@apache.org"
                  xmlns:xop="http://www.w3.org/2004/08/xop/include" />
                    </Payload>
                    <ErrorCode>Success</ErrorCode>
                    <ErrorMessage></ErrorMessage>
                </ns1:COREEnvelopeBatchSubmissionAckRetrievalResponse>
         </soapenv:Body>
  </soapenv:Envelope>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <1.urn:uuid:5117AAE1116EA8B87A1200060184692@apache.org>

<999 file>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339--
```

4.2.2.9 Batch Results Retrieval Request Message (non-normative) [19]

The Batch Results Retrieval Request message structure shown below specifies SOAP 1.2. The use of MTOM in Batch mode request/response creates multipart MIME even though there is no payload (which may be the case for a Batch Retrieval Request). This shows the following:

---

[18] Although this example shows a 999 payload type being sent as a response from a server to the client, this could also include a TA1. Alternatively, the server may elect to send only a TA1 without any functional group.

[19] The Generic Batch Retrieval Request (§6.3.3) uses the same request message as the Batch Results Retrieval Request message structure depicted below, with different *PayloadType* values based on what is being retrieved.

1. The HTTP Headers are shown colored in blue.
2. The WS-Security Username and Password token (shown here with a pink background) is added to the SOAP Header by the platform on which SOAP is run. The SOAP platform's Web-Services Security Extensions may be configured to insert these tokens.
3. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)

```
POST /core/eligibilityBatch HTTP/1.1
Content-Type:      multipart/related;      boundary=MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614;
type="application/xop+xml";     start="0.urn:uuid:5117AAE1116EA8B87A1200060184615@apache.org";     start-
info="application/soap+xml"; action="BatchResultsRetrievalTransaction"

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:5117AAE1116EA8B87A1200060184615@apache.org>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
       <soapenv:Header>
       <wsse:Security     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
       secext-1.0.xsd"                                                    soapenv:mustUnderstand="true">
       <wsse:UsernameToken           xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
       wssecurity-utility-1.0.xsd"                                      wsu:Id="UsernameToken-21621664">
       <wsse:Username>bob</wsse:Username>
       <wsse:Password       Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
       profile-1.0#PasswordText">bobPW</wsse:Password>
       </wsse:UsernameToken>
       </wsse:Security>
       </soapenv:Header>
       <soapenv:Body>
              <ns1:COREEnvelopeBatchResultsRetrievalRequest
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                <PayloadType>X12_005010_Request_Batch_Results_271</PayloadType>
                     <ProcessingMode>Batch</ProcessingMode>
                     <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                     <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                     <SenderID>HospitalA</SenderID>
                     <ReceiverID>PayerB</ReceiverID>
                     <CORERuleVersion>2.2.0</CORERuleVersion>
                </ns1:COREEnvelopeBatchResultsRetrievalRequest>
         </soapenv:Body>
  </soapenv:Envelope>
--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```


4.2.2.10 Batch Results Retrieval Response Message (non-normative) [20]

The Batch Results Retrieval Response message structure shown below specifies SOAP 1.2 and MTOM. This shows the following:

1. The HTTP Headers are shown colored in blue.
2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)
3. The MTOM Attachment is colored in orange.

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type:      multipart/related;      boundary=MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339;
type="application/xop+xml";     start="0.urn:uuid:0B72121B1FEFA9BDD31200060195340@apache.org";     start-
info="application/soap+xml"; action="BatchResultsRetrievalTransaction"

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:0B72121B1FEFA9BDD31200060195340@apache.org>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Body>
              <ns1:COREEnvelopeBatchResultsRetrievalResponse
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                    <PayloadType>X12_271_Response_005010X279A1</PayloadType>
                    <ProcessingMode>Batch</ProcessingMode>
                    <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                    <PayloadLength>1551254</PayloadLength>
                    <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                    <SenderID>PayerB</SenderID>
                    <ReceiverID>HospitalA</ReceiverID>
                    <CORERuleVersion>2.2.0</CORERuleVersion>
                    <CheckSum>43B8485AB5</CheckSum>
                  <Payload>
                  <xop:Include       href="cid:1.urn:uuid:5117AAE1116EA8B87A1200060184692@apache.org"
                  xmlns:xop="http://www.w3.org/2004/08/xop/include" />
                    </Payload>
                    <ErrorCode>Success</ErrorCode>
                    <ErrorMessage></ErrorMessage>
                </ns1:COREEnvelopeBatchResultsRetrievalResponse>
         </soapenv:Body>
  </soapenv:Envelope>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <1.urn:uuid:5117AAE1116EA8B87A1200060184692@apache.org>

<Response batch file>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339--
```

4.2.2.11 Batch Results Acknowledgement Submission Message (non-normative) [21]

The Batch Results Acknowledgement Submission message structure shown below specifies SOAP 1.2, and also uses MTOM (See §6.1) to send the payload file. This shows the following:

1. The HTTP Headers are shown colored in blue.

[20] The Generic Batch Retrieval Response (§6.3.3) uses the same response message as the Batch Results Retrieval Response message structure depicted below, with different *PayloadType* values based on the response to what is being retrieved.
[21] The Generic Batch Receipt Confirmation (§6.3.3) uses the same request message as the Batch Results Acknowledgement Submission message structure depicted below, with different *PayloadType* values as appropriate.

2. The WS-Security Username and Password token (shown here with a pink background) is added to the SOAP Header by the platform on which SOAP is run. The SOAP platform's Web-Services Security Extensions may be configured to insert these tokens.
3. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)
4. The Batch file (MTOM attachment) is shown colored in orange.

```
POST /core/eligibilityBatch HTTP/1.1
Content-Type:      multipart/related;      boundary=MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614;
type="application/xop+xml";     start="0.urn:uuid:5117AAE1116EA8B87A1200060184615@apache.org";     start-
info="application/soap+xml"; action="BatchResultsAckTransaction"

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:5117AAE1116EA8B87A1200060184615@apache.org>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
       <soapenv:Header>
       <wsse:Security     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
       secext-1.0.xsd"                                                    soapenv:mustUnderstand="true">
       <wsse:UsernameToken           xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
       wssecurity-utility-1.0.xsd"                                      wsu:Id="UsernameToken-21621664">
       <wsse:Username>bob</wsse:Username>
       <wsse:Password       Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
       profile-1.0#PasswordText">bobPW</wsse:Password>
       </wsse:UsernameToken>
       </wsse:Security>
       </soapenv:Header>
       <soapenv:Body>
              <ns1:COREEnvelopeBatchResultsAckSubmission
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                     <PayloadType>X12_999_SubmissionRequest_005010X231A1</PayloadType>
                     <ProcessingMode>Batch</ProcessingMode>
                     <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                     <PayloadLength>1551254</PayloadLength>
                     <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                     <SenderID>HospitalA</SenderID>
                     <ReceiverID>PayerB</ReceiverID>
                     <CORERuleVersion>2.2.0</CORERuleVersion>
                     <CheckSum>43B8485AB5</CheckSum>
                   <Payload>
                   <xop:Include       href="cid:1.urn:uuid:5117AAE1116EA8B87A1200060184692@apache.org"
                   xmlns:xop="http://www.w3.org/2004/08/xop/include" />
                     </Payload>
              </ns1:COREEnvelopeBatchResultsAckSubmission>
        </soapenv:Body>
  </soapenv:Envelope>

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <1.urn:uuid:5117AAE1116EA8B87A1200060184692@apache.org>

<999 file>

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```

4.2.2.12 Batch Results Acknowledgement Submission Response Message (non-normative) [22]

The Batch Results Acknowledgement Submission Response message structure shown below specifies SOAP 1.2 and MTOM. This shows the following:

[22] The Generic Batch Receipt Confirmation Response (§6.3.3) uses the same response message as the Batch Results Acknowledgement Submission Response message structure depicted below, with different *PayloadType* values as appropriate.

1. The HTTP Headers are shown colored in blue.
2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE Phase II Connectivity Rule. (See §4.4)

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type:  multipart/related;  boundary=MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339;
type="application/xop+xml"; start="0.urn:uuid:0B72121B1FEFA9BDD31200060195340@apache.org"; start-
info="application/soap+xml"; action="BatchResultsAckTransaction"

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339
Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"
Content-Transfer-Encoding: binary
Content-ID: <0.urn:uuid:0B72121B1FEFA9BDD31200060195340@apache.org>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Body>
              <ns1:COREEnvelopeBatchResultsAckSubmissionResponse
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                    <PayloadType>X12_Response_ConfirmReceiptReceived</PayloadType>
                      <ProcessingMode>Batch</ProcessingMode>
                      <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                      <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                      <SenderID>PayerB</SenderID>
                      <ReceiverID>HospitalA</ReceiverID>
                      <CORERuleVersion>2.2.0</CORERuleVersion>
                      <ErrorCode>Success</ErrorCode>
                      <ErrorMessage></ErrorMessage>
                 </ns1:COREEnvelopeBatchResultsAckSubmissionResponse>
         </soapenv:Body>
  </soapenv:Envelope>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339--
```

4.2.2.13 ErrorMessage Structure (non-normative)

The Error message structure shown below uses the SOAP Fault specifications within SOAP 1.2. As described in §4.3.3: Error Handling, SOAP Faults must be used to send errors at the SOAP level. The HTTP Headers are shown colored in blue. The remainder of the request is the SOAP Envelope.

```
HTTP/1.1 500
Content-Length: 2408
Content-Type: application/soap+xml


<soapenv:Envelope xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns: soapenv ="
http://www.w3.org/2003/05/soap-envelope/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

<soapenv:Header>

</soapenv:Header>

<soapenv:Body>

  <soapenv:Fault>
   <soapenv:Code><env:Value>env:Client</env:Value></env:Code>
    <soapenv:Reason>
      <soapenv:Text xml:lang="en">There was an error in the incoming SOAP request</env:Text>
    </soapenv:Reason>
   </soapenv:Fault>
</soapenv:Body>

</soapenv:Envelope>
```

4.2.2.14 Envelope Processing Error Message (non-normative)

The Error message structure shown below illustrates a SOAP-based message that indicates an error has occurred within processing the envelope. The HTTP Headers are shown colored in blue. The remainder of the request is the

SOAP Envelope. The envelope structure and metadata that is defined within CORE Phase II Connectivity Rule is colored in green.

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml;
action="http://www.caqh.org/SOAP/WSDL/CORETransactions/RealTimeTransactionResponse";charset=UTF-8

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
       <soapenv:Body>
              <ns1:COREEnvelopeRealTimeResponse xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule2.2.0.xsd">
                     <PayloadType>CoreEnvelopeError</PayloadType>
                     <ProcessingMode>RealTime</ProcessingMode>
                     <PayloadID>f81d4fae-7dec-11d0-a765-00a0c91e6bf6</PayloadID>
                     <TimeStamp>2007-08-30T10:20:34Z</TimeStamp>
                     <SenderID>PayerB</SenderID>
                     <ReceiverID>HospitalA</ReceiverID>
                     <CORERuleVersion>2.2.0</CORERuleVersion>
                     <Payload></Payload>
                     <ErrorCode>VersionMismatch</ErrorCode>
                     <ErrorMessage>Expecting Version X, received Version Y</ErrorMessage>
              </ns1:COREEnvelopeRealTimeResponse>
         </soapenv:Body>
  </soapenv:Envelope>
```

4.2.2.15 Payload Attachment Handling

Real Time

- Payload must be embedded using the Inline method (using CDATA element as shown above).

Batch

- Payload must be sent as an MTOM[23] encapsulated MIME part.

4.3 General Specifications Applicable to Both Envelope Methods

The following sections specify requirements of the CORE Phase II Connectivity Rule that are applicable to both envelope methods (HTTP MIME Multipart and SOAP+WSDL).

4.3.1 Request and Response Handling

HTTP/S supports a request-response message pattern, meaning that the sender submits a message and then waits for a response from the message receiver. This works well for the submission of both batch and real time ASC X12 messages, but the response message from the receiver is different depending on whether the sender's message is a real time request, batch submission, or batch request pickup.

4.3.1.1 Real Time Requests

Real time requests must include a single inquiry or submission (e.g., one eligibility inquiry to one information source for one patient). In this model the response from the message receiver is either an error response (see §4.3.3 Error Handling) or the corresponding ASC X12 message response (e.g., a TA1, v5010 999 or v5010 271 if the request was a v5010 270).

4.3.1.2 Batch Submission

Batch requests are sent in the same way as real time requests. Batch requests are acknowledged using an HTTP/S acknowledgement and the envelope standard and metadata as specified in §4.4.

4.3.1.3 Batch Response Pickup

Batch responses are retrieved after the message receiver has had a chance to process a batch submission (see the CORE 155: Batch Response Time Rule version 1.1.0 and CORE 250 Claim Status Rule version 2.1.0, §4.5: Claim Status Response Time Requirements for details on timing.) Under this usage pattern, the message sender connects to the message

[23] MTOM is defined in Appendix §6.1: *Definitions and Abbreviations used in this Rule.*

receiver using HTTP/S with the envelope standard and metadata specified in §4.4, and sends a message requesting available files. The responder then sends back either:

1. The file(s) in the HTTP/S response message (payload) using the envelope standard and metadata specified in §4.4, or
2. A list of available file(s) in the HTTP/S response message (payload), and supports a mechanism to request a particular file from the list. If a Batch response contains a list of files, then the site's Connectivity Companion Guide (See §4.3.7) will specify the method to pick up each file in the list.

4.3.2 Submitter Authentication and Authorization Handling

The two methods for Submitter Authentication specified in this rule are:

1. Username/Password (Referred to as Submitter Authentication Standard C in the Conformance Requirements, §4.1) using the CORE-compliant Envelope to send the *UserName* and *Password*, as specified in the CORE-compliant Envelope Metadata specifications. For both envelope methods, the Username and Password are a part of the CORE-compliant envelope. For the SOAP+WSDL method, the WS-Security standard must be used to embed the Username and Password values inside the Envelope, as illustrated in the examples above.
2. X.509 Certificate based authentication over SSL[24] (Submitter Authentication Standard D in the Conformance Requirements, §4.1), using the Secure Sockets Layer (SSLv3.0) open standard for client certificate based authentication.

The submitter authentication conformance requirements for stakeholders are defined in §4.1.

Submitter Authorization is assumed to be a local decision at the site that receives the submission.

4.3.3 Error Handling

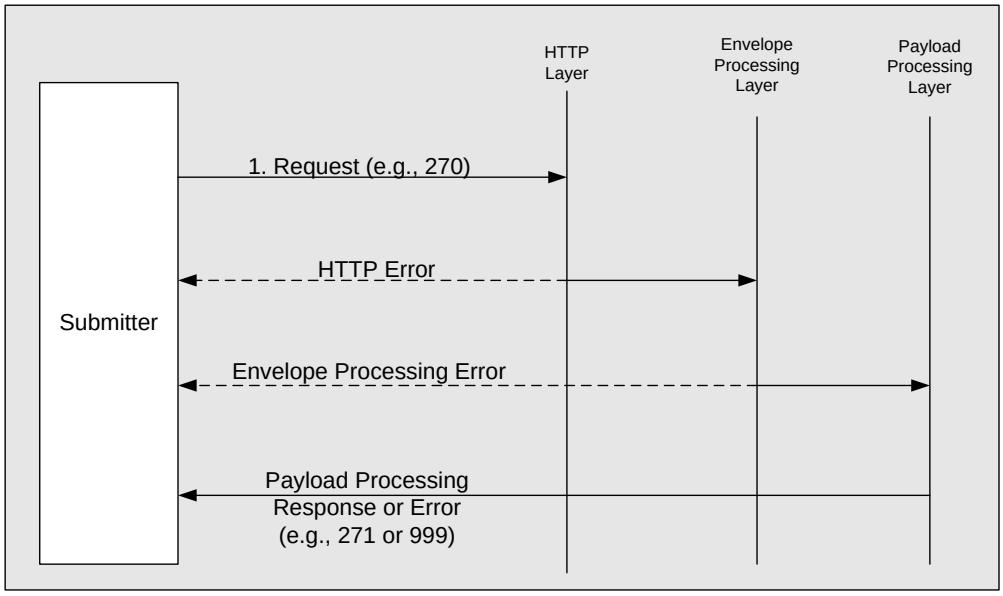
The error handling described in this section is applicable to both envelope methods. As shown in Figure #4.3.3 below, a submitted request goes through at least three logical layers that process the request.

- Processing of HTTP headers (typically handled by a web-server)
- Processing the Envelope (can be handled by messaging middle-ware or integration brokers)
- Processing the Payload (e.g., ASC X12, typically handled by application business logic)

Once a request (e.g., ASC X12 v5010 270) is submitted it goes through these 3 logical layers. At each of these layers, some part of the request is processed. At each layer there can be errors (indicated by the dotted arrows being returned to the request submitter), which may be returned to the request submitter. If there is an error in processing the message at any logical layer, the request does not get passed to the next layer. If no errors are encountered at that layer, the request is passed to the next processing layer. The last logical layer that processes the request is the Payload Processing Layer. Once this layer processes the payload, it returns a response or error (e.g., ASC X12 v5010 271 or v5010 999 or TA1).

[24] Reference §3.1 for more information regarding the use of SSL and the optional use of TLS 1.0.

Figure #4.3.3



Note: In Figure #4.3.3 above, the dotted line arrows indicate error messages being returned to the Submitter if there is a processing error at the corresponding logical processing layer. The straight line arrows indicate the request and response messages.

4.3.3.1 HTTP Status and Error Codes (Normative, Not Comprehensive[25])

The processing and error codes for the HTTP Layer are defined as part of the HTTP specifications [*http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html*]. The intended use of these status and error codes in processing the requests are specified in Table 4.3.3.1 and are consistent with the HTTP status codes from CORE Phase I.

Note: The status and error codes included in Table 4.3.3.1 only represent a short list of several commonly used status codes in the standard. An exhaustive list of HTTP Status Codes and descriptions are included in the HTTP specification [http://tools.ietf.org/html/rfc2616#section-6.1.1] Phase II CORE Connectivity requires the use of the appropriate HTTP error or status codes as applicable to the error/status situation.

Table 4.3.3.1

HTTP Status/Error Codes (Normative, Not Comprehensive)	**Status Code Description[26] (Intended Use)**
200 OK	Success
202 Accepted	Batch file submission has been accepted (but not necessarily processed)
400 Bad Request	Incorrectly formatted HTTP headers

[25] An exhaustive list of HTTP Status Codes and definitions are included in section 6.1.1 of the HTTP specification [http://tools.ietf.org/html/rfc2616#section-6.1.1].
[26] Section 6.1.1 of the HTTP specification http://tools.ietf.org/html/rfc2616#section-6.1.1.

HTTP Status/Error Codes (Normative, Not Comprehensive)	Status Code Description[26] (Intended Use)
403 Forbidden	Access denied (e.g., X.509 client certificate based authentication)[27]
500 Internal Server Error	The web-server encountered a processing error, or there was a SOAP fault (in case of SOAP envelope method)
5xx Server errors	Standard set of server side errors (e.g., 503 Service Unavailable)

4.3.3.2 Envelope Processing Status and Error Codes (Normative, Comprehensive)

When SOAP is used, some of the CORE-compliant Envelope Processing errors map to SOAP faults [*http://www.w3.org/TR/soap12-part1/#soapfault*]. (See Table 4.3.3.2) To handle CORE-compliant envelope processing status and error codes, two fields called *ErrorCode* and *ErrorMessage* are included in the CORE-compliant Envelope. (See §4.4) *ErrorMessage* is a free form text field that describes the error (for the purpose of troubleshooting/logging). When an error occurs, *PayloadType* is set to *CoreEnvelopeError*.

Table 4.3.3.2

CORE-compliant Envelope Processing Status/Error Codes (Normative, Comprehensive)	Status Code Description[28] (Intended Use)
Success	Envelope was processed successfully.
<FieldName>Illegal	Illegal value provided for <FieldName>.
<FieldName>Required	The field <FieldName> is required but was not provided.
<FieldName>NotUnderstood	The field <FieldName> is not understood at the receiver. In the case of SOAP, this error is returned as a NotUnderstood SOAP fault.
VersionMismatch	The version of the envelope sent is not acceptable to the receiver. If the SOAP version is not valid at the receiver, a SOAP fault is returned with this fault code.
UnAuthorized	The username/password or Client certificate could not be verified.
ChecksumMismatched	The checksum value computed on the recipient did not match the value that was sent in the envelope.
Sender	The envelope sent by the sender did not conform to the expected format. In the case of SOAP, this error should be sent as a SOAP fault with "Sender" fault code.

[27] If the username/password option is used, then these values are specified in the CORE-compliant envelope, which is not processed at the HTTP layer. This is handled by the error code (Unauthorized.)
[28] An exhaustive list of HTTP Status Codes and definitions are included in section 6.1.1 of the HTTP specification [http://tools.ietf.org/html/rfc2616#section-6.1.1].

CORE-compliant Envelope Processing Status/Error Codes (Normative, Comprehensive)	Status Code Description[28] (Intended Use)
Receiver	The message could not be processed for reasons attributable to the Receiver (e.g., upstream process is not reachable). In the case of SOAP, this error should be sent as a SOAP fault with "Receiver" fault code.

4.3.3.3 Examples of Status and Error Codes (non-normative)

The following illustrates the status and error codes that may be returned:

- A SOAP or MIME Multipart request that has illegal HTTP headers gets a response with HTTP Error Code: "400 Bad Request."
- A SOAP or MIME Multipart request with an unauthorized submitter's client-certificate (in case that method is being used) gets a response with HTTP Error Code: "403 Forbidden."
- A SOAP request with HTTP headers properly formatted but using the wrong SOAP version (1.1 instead of 1.2) gets HTTP Status 500, and a SOAP fault with a fault code "VersionMismatch."
- A MIME Multipart request with HTTP headers properly formatted, uses a properly formatted MIME Header but does not have *PayloadType* in the CORE-compliant envelope gets a HTTP Status 200 (since it passed the HTTP layer successfully), but receives an Envelope processing error with *ErrorCode* set to *PayloadTypeRequired*.
- A MIME Multipart request with HTTP headers properly formatted uses a properly formatted MIME Header but has *PayloadType* in the CORE-compliant envelope with an unknown value (i.e., not in the Coded Set defined for this field). This gets a HTTP Status 200 response (since it passed the HTTP layer successfully), but receives an Envelope processing error with *ErrorCode* set to *PayloadTypeIllegal.*

4.3.3.4 Examples of Error Messages (non-normative)

ErrorMessage field is intended to provide a descriptive text of the error message in free form text, to aid in logging and troubleshooting. It is the responsibility of the implementer to keep this message consistent with the semantics of the *ErrorCode*, and not in conflict with it. The *ErrorMessage* must be related to the *ErrorCode* as defined in the table above. The following illustrates *ErrorMessage* fields that may be returned:

- For *ErrorCode = VersionMismatch*, the *ErrorMessage* could be "Expecting CORERuleVersion=X, Received CORERuleVersion=Y"
- For *ErrorCode = SenderIDIllegal*, the *ErrorMessage* could be "SenderID length exceeds maximum allowed length"
- For *ErrorCode = TimeStampIllegal*, the *ErrorMessage* could be "Timestamp is missing the time-zone information"
- For *ErrorCode = CheckSumIllegal*, the *ErrorMessage* could be "Unknown algorithm", or "Unknown encoding type"
- For *ErrorCode = Receiver*, the *ErrorMessage* could be "Failed to connect to backend system X to process this message"

4.3.4 Audit Handling

Auditing is a local decision by each trading partner. The CORE recommended best practice is for each trading partner to audit all the envelope metadata and payload for each transaction.

4.3.4.1 Tracking of Date and Time and Payload ID

In order to comply with the Phase I CORE 155 and 156: Response Time Rules version 1.1.0 and Phase II CORE 250 Claim Status Rule, version 2.1.0, §4.4 and §4.5: Claim Status Response Time Requirements, message receivers will be required to track the times of any received inbound messages, and respond with the outbound message for that payload ID. In addition, as specified in the CORE Envelope Metadata Table 4.4.2, message senders must include the date and time the message was sent in the CORE metadata element Time Stamp.

4.3.5 Capacity Plan

4.3.5.1 Real Time Transactions

A CORE-certified entity must have a capacity plan such that it can receive and process a large number of single concurrent real time transactions via an equivalent number of concurrent connections. These single transactions must be received, processed and the appropriate response provided back to the sender within response time requirements specified in Phase I CORE 156 Real Time Response Time Rule version 1.1.0 and Phase II CORE 250 Claim Status Rule version 2.1.0, §4.4: Real Time Response Time Requirements.

Three major factors affect the specific number of Large Volume of Single Real time Transactions (See §6.1) capable of being transported and processed within a given CORE response time frame. They are:

1. The amount of message metadata and message encapsulation structure which is required for each transaction;
2. The characteristics of the message handling software and how concise its design and coding are; and,
3. The architecture of the intervening hardware, software and communication platform.

CORE-certified entities must attest that their capacity planning addresses the above 3 factors that affect large volume single real time processing[29]. CORE-certified entities must also attest that they have the ability to track, on a calendar week basis, any change to their agreed upon volume capacity.

In the circumstances where the transaction volume throughput is exceeded by one of the trading partners, the receiving organization may declare a denial of service event and request a temporary waiver of the applicable CORE response time rule's performance criteria, and/or other appropriate action.

4.3.5.2 Batch Transactions

The CORE-certified messaging system must have the capability to receive and process large batch transaction files if the entity supports batch transactions. These transactions must be received, processed and the appropriate response provided back to the sender within the time specified in the applicable CORE Rule.

Three major factors that affect the specific number of Large Batch payloads capable of being transported and processed within a given time frame are:

1. The availability and use of capabilities in the messaging protocol which support in-line files, file attachments, and automated integrity assurance routines, etc., together with the quality and characteristics of their implementation;
2. The characteristics of the message handling software and its conciseness of design and coding; and,
3. The architecture of the intervening hardware, software and communication platform.

CORE-certified entities must attest that their capacity planning addresses the above 3 factors that affect large batch processing. The maximum number of transaction sets to be included in a large batch file is determined between trading partners.

[29] See *Appendix 6.1: Abbreviations and Definitions used in this Rule* for a definition of Large Volume of Single Real time Transactions (Synchronous).

4.3.6 Response, Timeout and Retransmission Requirements

- Real time response must conform to Phase I CORE 156 Eligibility and Benefits Real Time Response Time Rule version 1.1.0.
- Batch response time must conform to Phase I CORE 155 Eligibility and Benefits Batch Response Time Rule version 1.1.0.

If a Real time response message is not received within the 60 second response period, the submitter's system should send a duplicate transaction no sooner than 90 seconds after the original attempt was sent.

If no Real time response is received after the second attempt, the submitter's system should submit no more than 5 duplicate transactions within the next 15 minutes.

If additional attempts result in the same timeout termination, the submitter's system must notify the submitter to contact the receiver directly to determine if system availability problems exist or if there are known Internet traffic constraints causing the delay.

4.3.7 Publication of Entity-Specific Connectivity Guide

Servers must publish detailed specifications in a Connectivity Companion Guide on the entity's public web site. CORE recommends specifying the following:

- CORE Rule Version for Connectivity.
- Details on the message format and the supported transactions (e.g., Real time, Batch transactions).
- Details about the entity's ASC X12 Interchange; e.g., will an interchange contain multiple functional groups; will the TA1 be in its own interchange without any functional group(s).
- An entity may implement custom extensions which are not compliant with the CORE Connectivity Rule. Any such extension must be specified in the entity's companion guide. Consistent with the CORE Safe Harbor principle (§5 Safe Harbor), a CORE certified entity can implement custom extensions and/or support additional connectivity methods as long as the organization has implemented one connectivity interface that is fully and exactly as specified in the CORE Connectivity Rule. This gives CORE certified partners the assurance that they can use their CORE Connectivity-certified interfaces.
- If a Batch response contains a list of files (instead of returning the Batch file itself), the location and mechanism to pick up each file in that list (e.g., FTP, HTTP, or Web-Service) should be specified in the entity's Connectivity Companion Guide.. When a Batch response contains a list of files, it is an example of a custom extension that is considered outside of the CORE Safe Harbor.
- Value of *ReceiverID* for that site.
- Production and Testing URLs for Real time and Batch transactions.
- Maximum number of Real time and Batch transactions that can be sent per minute by a single trading partner (Client).
- Maximum size of Batch files that can be received by a Server.
- Authentication/Authorization policies using either X.509 Client Certificates or User ID and Password (e.g., how to enroll and obtain a Client Certificate or *UserID* and *Password* to connect to that Receiver).
- Information on obtaining the Receiver's Root Certificate Authority and/or Intermediate Certificate Authority public key certificate.
- System Availability as required by Phase I CORE 157 System Availability Rule version 1.1.0 and Phase II CORE 250 Claim Status Rule version 2.1.0, §4.6: Claim Status System Availability.
- Business/Technical points of contact.
- Rules of behavior for programs that connect to this site (e.g., must not deliberately submit batch files that contain Viruses).

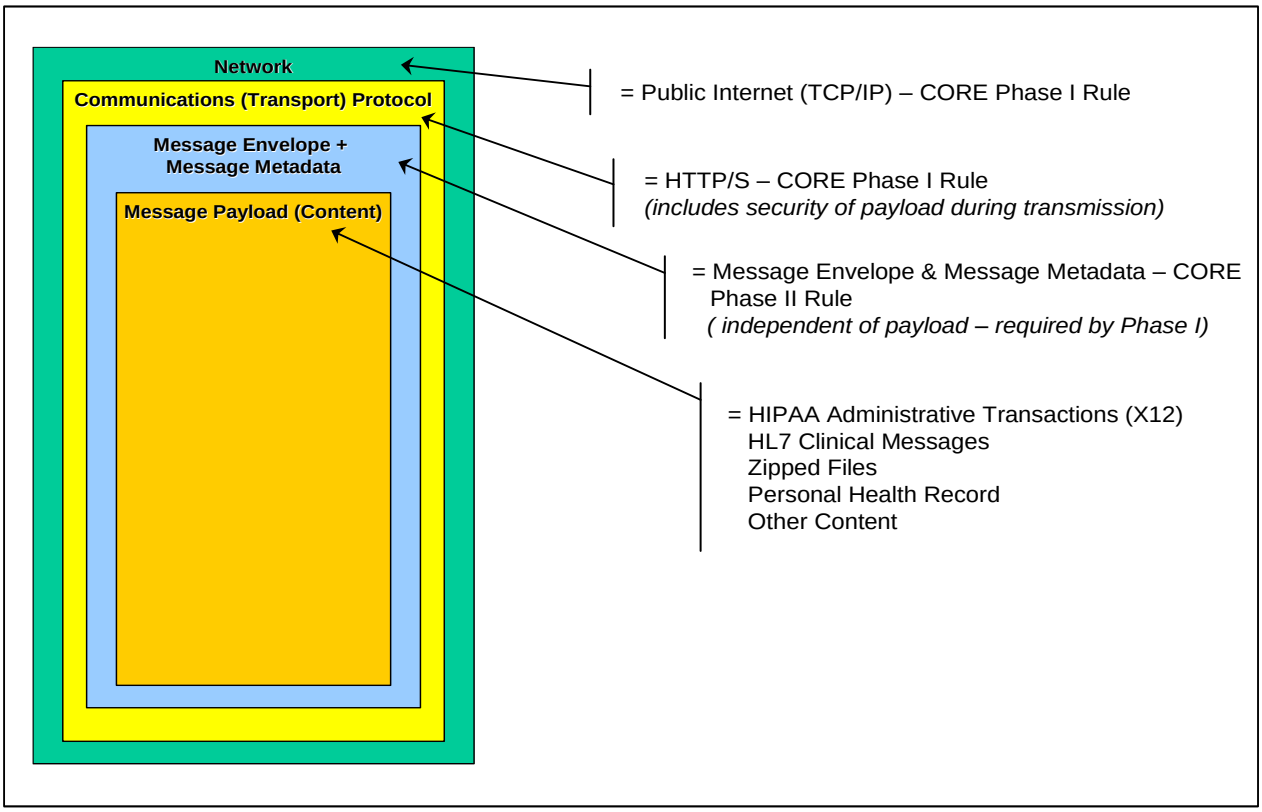
4.4 Envelope Metadata Fields, Descriptions, Intended Use and Syntax/Value-Sets

The Envelope Metadata specified in Table 4.4.2 on the following page pertains to the Phase II Message Envelope identified in §4.4.1, and is applicable to both enveloping standards (SOAP+WSDL and HTTP MIME Multipart) that may be used for encoding the Phase II message envelope (subject to the conformance requirements discussed in the Conformance section of this rule). With the exception of *ErrorCode* and *ErrorMessage* fields, which are only sent in the response, the CORE Phase II-required envelope metadata for the request and response are required to be identical.

4.4.1 Message Envelope

As shown in Figure #4.4.1 below, the Message Envelope is outside the Message Payload (content), and inside the transport protocol envelope. The Phase I CORE 153 Connectivity Rule version 1.1.0 was based on the use of HTTP/S as the transport protocol; hence the transport protocol envelope consists of HTTP headers. Examples of message payload include HIPAA administrative transactions (ASC X12), HL7 clinical messages, zipped files, etc.

Figure #4.4.1



4.4.2 Table of CORE Envelope Metadata

Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[30]**	**Requirement Indicator for Real Time and Batch**	**Data Type**	**Field Constraints or Value-sets (Not comprehensive)**
Payload Type	Payload Type specifies the type of payload included within a request, (e.g. HIPAA X12 transaction set 270, 276, 278, etc.).	• Message routing • Efficient processing • Auditing	PayloadType	Required for both	Coded Set	Please see §4.4.3 for enumeration of PayloadType field.
Processing Mode	Processing Mode indicates Batch or Real time[31] processing mode (as defined by CORE)	• Messaging routing • Resource allocation • Transaction scheduling • Message or transaction auditing	ProcessingMode	Required for both	Coded Set	RealTime, Batch
Payload Length	Defines the length of the actual payload in bytes.	• Efficient processing and resource allocation. • Auditing • Trouble-shooting	PayloadLength	Required for Batch interactions except under certain conditions[32] Shall not be used for Real time.	Integer (Base 10)	
Payload ID	Payload ID (unique within the domain of the party that sets this value) is a payload identifier assigned by the Sender in both Batch and Real Time processing modes. If the payload is being resent in the absence of confirmation of receipt to persistent storage, the same PayloadID may be re-used.	• Auditing • Trouble-shooting	PayloadID	Required for both Real time and Batch.	String	*PayloadID* will conform to ISO UUID standards (described at ftp://ftp.rfc-editor.org/in-notes/rfc4122.txt), with hexadecimal notation, generated using a combination of local timestamp (in milliseconds) as well as the hardware (MAC) address[33], to ensure uniqueness.

[30] Mixed case (e.g., *PayloadType*) is used for the field names, since this is consistent with the WS-Security tags that are used for authentication.
[31] See *Appendix 6.1: Abbreviations and Definitions used in this Rule* for a definition of Batch and Real time.
[32] Some requests or responses within Batch interactions may not have a payload. This could occur when requesting a payload or when there is no payload in the response.
[33] In multithreaded environments, in addition to the hardware (MAC) address and timestamp, the Process-ID or Thread-ID may also be used as additional parameters to ensure *PayloadID* uniqueness across multiple processes and/or threads.

Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[30]**	**Requirement Indicator for Real Time and Batch**	**Data Type**	**Field Constraints or Value-sets (Not comprehensive)**
Time Stamp (Adjusted from Phase I)	The Sender (request) or Receiver (response) Time Stamp combines Phase I time and date[34] message metadata into a single Coordinated Universal Time (UTC) time stamp (including time zone information) specifying when a message is created and sent to a receiver. This does not require a shared time server for consistent time.	• Auditing • Trouble-shooting	TimeStamp	Required for both	dateTime	dateTime (http://www.w3.org/TR/xmlschema11-2/#dateTime)
User Name[35] (Already in Phase I)	User Name is part of User Name/Password based authentication credentials. It can also be used by the receiver for authorization purposes. User Name is protected by transport layer SSL[36] encryption. User Name must be used only for identification, authentication and authorization purposes while Sender Identifier must be used to identify trading partners and convey other business information within a transaction.	• Authentication and authorization • Auditing	UserName	Required for both if X.509 Client-certificate authentication over SSL/TLS[37] is not used.	String	Maximum length 50 characters
Password (Already in Phase I)	Password is part of the User Name/Password based authentication credentials. Password is protected by transport layer SSL[38] encryption.	Submitter Authentication	Password	Required for both if X.509 Client-certificate authentication over SSL/TLS is not used. If UserName field is present, a corresponding Password must be present.	String	Maximum length 50 characters

[34] See CORE Phase I Connectivity Rule for time, date message metadata requirements.
[35] See CORE Phase I Connectivity Rule for User ID and Password as part of the Security and Authentication Requirements.
[36] Reference §3.1 for more information regarding the use of SSL and the optional use of TLS 1.0.
[37] This type of authentication is consistent with the IHE's Audit Trail and Node Authentication (ATNA) profile.
[38] Reference §3.1 for more information regarding the use of SSL and the optional use of TLS 1.0.

Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[30]**	**Requirement Indicator for Real Time and Batch**	**Data Type**	**Field Constraints or Value-sets (Not comprehensive)**
Sender Identifier	A unique[39] business entity identifier representing the message envelope creator. Sender Identifier is better suited for identifying business entities and trading partners than User Name because: • User Name is usually anonymized for security reasons and to protect privacy. • User Name attribute does not exist if another authentication method is used. • Authentication and messaging may happen on different layers[40] and therefore may be handled by disparate applications and processes.	• Message routing and processing by a receiver • Transaction auditing. • As a reference to a business agreement.	SenderID	Required	String	Maximum length 50 characters The use of OIDs (e.g., HL7 or IANA) is recommended, but not required.
Receiver Identifier	A unique[41] business entity identifier representing the next-hop receiver.	• Transaction auditing. • As a reference to a business agreement. • Message routing by the receiver.	ReceiverID	Required	String	Maximum length 50 characters The use of OIDs (e.g., HL7 or IANA) is recommended, but not required.
CORE Rule Version	The CORE Rule version that this envelope is using. This value can be used to maintain backward compatibility when parsing/processing messages. A Phase II certified receiver must be able to interact with both Phase I and Phase II certified senders for backward compatibility.	• Message routing and processing. • Auditing • Backward compatibility.	CORERuleVersion	Required for both	Coded Set	2.2.0

[39] Unique within the Sender's domain.
[40] §2 shows the layers in the OSI model.
[41] Unique within a Receiver's domain.

Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[30]**	**Requirement Indicator for Real Time and Batch**	**Data Type**	**Field Constraints or Value-sets (Not comprehensive)**
Checksum	An element used to allow receiving site to verify the integrity of the message that is sent.	Message Integrity verification	CheckSum	Required for Batch interactions except under certain conditions[42] Not used for Real time	String	Algorithm is SHA-1, Encoding is Hex. Checksum must be computed only on the payload and not on the metadata.
Error Code	Error code to indicate the error when processing the envelope.	• Error handling • Troubleshooting	ErrorCode	Required in Response (for both Real time and Batch) Not used in Request.	Coded Set	Please see Section on Error Handling for a definition of error codes.
Error Message	Text Error message that describes the condition that caused the error. The text of the *ErrorMessage* must provide additional information describing how the Error can be resolved, and must not provide conflicting information from that provided in the *ErrorCode*.	• Logging • Troubleshooting	ErrorMessage	Required in Response (for both Real time and Batch) Not used in Request	String	Maximum length of 1024 characters. Please see Section on Error Handling for examples of Error Messages.

[42] Some requests or responses within Batch interactions may not have a payload. This could occur when requesting a payload or when there is no payload in the response.

4.4.3 Enumeration of Processing Mode and PayloadType Fields

4.4.3.1 Real Time Transactions

For Real time requests, *ProcessingMode* must be set to *RealTime*, and *PayloadType* must be set to the following values for Request, Response and Errors.

Payloads to Support All ASC X12 Administrative Transactions for Real Time					
Transaction Name	**HIPAA Mandated**	**Payload Type Value (Normative/Comprehensive)**			
		V4010A1[43]		**V5010**	
		Request	**Response**	**Request**	**Response**
Health Care Eligibility Benefit Inquiry and Response	Y	X12_270_Request_004010X092A1	X12_271_Response_004010X092A1	X12_270_Request_005010X279A1	X12_271_Response_005010X279A1
Health Care Claim Status Request and Response	Y	X12_276_Request_004010X093A1	X12_277_Response_004010X093A1	X12_276_Request_005010X212	X12_277_Response_005010X212
Health Care Services Request for Review and Response	Y	X12_278_Request_004010X094A1	X12_278_Response_004010X094A1	X12_278_Request_005010X217E1_2	X12_278_Response_005010X217E1_2
Health Care Services Review - Inquiry & Response	N	X12_278_Request_004010X059	X12_278_Response_004010X059	X12_278_Request_005010X215	X12_278_Response_005010X215
Health Care Services Review - Notification & Announcement	N	X12_278_Request_004010X111	X12_278_Response_004010X111	X12_278_Request_005010X216E2	X12_278_Response_005010X216E2
Real Time Adjudication					
Health Care Claim: Institutional	N			X12_837_ Request_005010X223A1_2	X12_835_ Response_005010X221A1
Health Care Claim: Professional	N			X12_837_ Request_005010X222A1	X12_835_ Response_005010X221A1
Health Care Claim: Dental	N			X12_837_ Request_005010X224A1_2	X12_835_ Response_005010X221A1

[43] Several PayloadType values referencing version 004010 of various administrative transactions are being retained since these implementation guides are not adopted under HIPAA. As such, the industry may continue using these implementation specifications to support their business needs and are not required to move to a 5010 version of them. The corresponding v5010 implementation guides have been added to the value set to support the industry's transition to v5010 in general.

Payloads to Support All ASC X12 Administrative Transactions for Real Time					
Transaction Name	**HIPAA Mandated**	**Payload Type Value (Normative/Comprehensive)**			
		V4010A1[44]		**V5010**	
		Request	**Response**	**Request**	**Response**
Acknowledgements					
TA1 Interchange Acknowledgement	N		X12_TA1_Response_00401[45]		X12_TA1_Response_00501X231A1[46]
Functional Acknowledgement	N		X12_997_Response_004010		
Implementation Acknowledgement	N				X12_999_Response_005010X231A1
Errors					
CORE Envelope Error	N/A		CoreEnvelopeError		CoreEnvelopeError

[44] Several PayloadType values referencing version 004010 of various administrative transactions are being retained since these implementation guides are not adopted under HIPAA. As such, the industry may continue using these implementation specifications to support their business needs and are not required to move to a 5010 version of them. The corresponding v5010 implementation guides have been added to the value set to support the industry's transition to v5010 in general.

[45] The use of the TA1 Interchange Acknowledgement is not specified in a separate ASC X12 Implementation Specification. Reference Appendix B EDI Control Directory of the ASC X12N v4010 Implementation Guide for technical details of the TA1. Reference Appendix C EDI Control Directory of the ASC X12C 005010231A1 Implementation Acknowledgement for Health Care Insurance (999) Technical Report Type 3 for implementation guidance of the TA1 with v5010 ASC X12 administrative transactions.

[46] Ibid.

4.4.3.2 Batch Transactions

For Batch requests, *ProcessingMode* must be set to *Batch*, and *PayloadType* must be set to the following values for Request, Response and Errors.

Payloads to Support All ASC X12 Administrative Transactions for Batch					
Transaction Name	**HIPAA Mandated**	**Payload Type Value (Normative/Comprehensive)**			
		V4010A1[47]		**V5010**	
		Request	**Response**	**Request**	**Response**
Batch Payload (Submission and Subsequent Batch Response Pick Up) See §6.3.2 Batch Interaction					
Health Care Eligibility Benefit Inquiry and Response	Y	Batch Submission: X12_270_Request_004010X092A1 Results Retrieval Request: X12_004010_Request_BatchResults271	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_271_Response_004010X092A1	Batch Submission: X12_270_Request_005010X279A1 Results Retrieval Request: X12_005010_Request_Batch_Results_271	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_271_Response_005010X279A1
Health Care Claim Status Request and Response	Y	Batch Submission: X12_276_Request_004010X093A1 Results Retrieval Request: X12_004010_Request_BatchResults277	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_277_Response_004010X093A1	Batch Submission: X12_276_Request_005010X212 Results Retrieval Request: X12_005010_Request_Batch_Results_277	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_277_Response_005010X212
Health Care Services Request for Review and Response	Y	Batch Submission: X12_278_Request_004010X094A1 Results Retrieval Request: X12_278_Request_Batch_Results_004010X059	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_278_Response_004010X094A1	Batch Submission: X12_278_Request_005010X217E1 Results Retrieval Request: X12_278_Request_Batch_Results_005010X217E1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_278_Response_005010X217E1
Health Care Services Review - Inquiry & Response	N	Batch Submission: X12_278_Request_004010X059 Results Retrieval Request: X12_278_Request_Batch_Results_004010X059	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_278_Response_004010X059	Batch Submission: X12_278_Request_005010X215E1 Results Retrieval Request: X12_278_Request_Batch_Results_005010X215E1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_278_Response_005010X215E1
Health Care Services Review - Notification & Announcement	N	Batch Submission: X12_278_Request_004010X111 Results Retrieval Request: X12_278_Request_Batch_Results_004010X111	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_278_Response_004010X111	Batch Submission: X12_278_Request_005010X216E1_2 Results Retrieval Request: X12_278_Request_Batch_Results_005010X216E1_2	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Results Retrieval Response: X12_278_Response_005010X216E1_2
Batch Payload (Pick Up only) See §6.3.3 Generic Batch Retrieval Request and Asynchronous Receipt Confirmation					
Health Care Claim Acknowledgement	N			Retrieval Request: X12_277CA_Request_005010X214E1_2	Retrieval Response: X12_277CA_Response_005010X214E1_2
Health Care Claim Payment/Advice	Y	Retrieval Request: X12_835_Request_004010X091A1	Retrieval Response: X12_835_Response_004010X091A1	Retrieval Request: X12_835_Request_005010X221A1	Retrieval Response: X12_835_Response_005010X221A1

[47] Several PayloadType values referencing version 004010 of various administrative transactions are being retained since these implementation guides are not adopted under HIPAA. As such, the industry may continue using these implementation specifications to support their business needs and are not required to move to a 5010 version of them. The corresponding v5010 implementation guides have been added to the value set to support the industry's transition to v5010 in general.

Payloads to Support All ASC X12 Administrative Transactions for Batch					
Transaction Name	**HIPAA Mandated**	**Payload Type Value (Normative/Comprehensive)**			
		V4010A1[47]		**V5010**	
		Request	**Response**	**Request**	**Response**
Health Care Claim Pending Status Information	N			Retrieval Request: X12_277_Request_005010X228E1	Retrieval Response: X12_277_Response_005010X228E1
Batch Payload (Submission of Payload followed by Submission of Results) See §6.3.4 Generic Batch Submission and Synchronous Receipt Confirmation					
Health Care Claim Request for Additional Information	N			Batch Submission (step 1): X12_277_Request_005010X213E1_2 Batch Submission (step 2): X12_275_Request_005010X210E1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Acknowledgement: X12_TG1_Response_00501X231A or X12_999_Response_005010X231A1
Request for Information in Support of a Disability Claim	N			Batch Submission (Step 1): X12_277_Request_005010X227 Batch Submission (Step 2) X12_275_Request_005010X210E1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation Acknowledgement: X12_TG1_Response_00501X231A or X12_999_Response_005010X231A1
Batch Payload (Submission only) See §6.3.4 Generic Batch Submission and Synchronous Receipt Confirmation					
Health Care Claim: Institutional	Y	Batch Submission: X12_837_Request_004010X096A1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation	Batch Submission: X12_837_Request_005010X223A1_2	Batch Receipt Confirmation: X12_BatchReceiptConfirmation
Health Care Claim: Professional	Y	Batch Submission: X12_837_Request_004010X098A1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation	Batch Submission: X12_837_Request_005010X222A1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation
Health Care Claim: Dental	Y	Batch Submission: X12_837_Request_004010X097A1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation	Batch Submission: X12_837_Request_005010X224A1_2	Batch Receipt Confirmation: X12_BatchReceiptConfirmation
Payroll Deducted and Other Group Premium Payment for Insurance Products	Y	Batch Submission: X12_820_Request_004010X061A1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation	Batch Submission: X12_820_Request_005010X218A1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation
Benefit Enrollment and Maintenance	Y	Batch Submission: X12_834_Request_004010X095A1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation	Batch Submission: X12_834_Request_005010X220A1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation
Health Care Benefit Coordination Verification	N			Batch Submission: X12_269_Request_005010X187	Batch Receipt Confirmation: X12_BatchReceiptConfirmation
Health Care Predetermination – Professional	N			Batch Submission: X12_837_Request_005010X291	Batch Receipt Confirmation: X12_BatchReceiptConfirmation
Health Care Predetermination – Institutional	N			Batch Submission: X12_837_Request_005010X292	Batch Receipt Confirmation: X12_BatchReceiptConfirmation
Doctors First Report of Injury	N	Batch Submission: X12_148_Request_004010X148	Batch Receipt Confirmation: X12_BatchReceiptConfirmation		

Payloads to Support All ASC X12 Administrative Transactions for Batch					
Transaction Name	**HIPAA Mandated**	**Payload Type Value (Normative/Comprehensive)**			
		V4010A1[47]		**V5010**	
		Request	**Response**	**Request**	**Response**
Health Care Service: Data Reporting	N			Batch Submission: X12_837_Request_005010X225A1_2E1	Batch Receipt Confirmation: X12_BatchReceiptConfirmation

Payloads to Support All ASC X12 Administrative Transactions for Batch					
Transaction Name	**HIPAA Mandated**	**Payload Type Value (Normative/Comprehensive)**			
		V4010A1[48]		**V5010**	
		Request	**Response**	**Request**	**Response**
Mixed/Non-Payload Specific Batch See §6.3.2 Batch Interaction					
Batch Submission (mixed payload types)	N/A	X12_004010_Request_BatchSubmissionMixed	X12_004010_Response_BatchSubmissionMixed	X12_005010_Request_BatchSubmissionMixed	X12_005010_Response_BatchSubmissionMixed
Batch Results Retrieval (mixed payload types)	N/A	X12_004010_Request_BatchResultsMixed	X12_004010_Response_BatchResultsMixed	X12_005010_Request_BatchResultsMixed	X12_005010_Response_BatchResultsMixed
No Batch Results Available	N/A		X12_004010_Response_NoBatchResultsFile		X12_005010_Response_NoBatchResultsFile
Acknowledgements					
Batch Receipt Confirmation Response	N		X12_BatchReceiptConfirmation		X12_BatchReceiptConfirmation
TA1 Interchange Acknowledgement Submission	N	X12_TA1_SubmissionRequest_00401[49]	X12_Response_ConfirmReceiptReceived	X12_TA1_SubmissionRequest_00501X231A1[50]	X12_Response_ConfirmReceiptReceived
Retrieval of TA1	N	X12_TA1_RetrievalRequest_00401	X12_TA1_Response_00401[51]	X12_TA1_RetrievalRequest_00501X231A1	X12_TA1_Response_00501X231A1[52]
997 Functional Acknowledgement Submission	N	X12_004010_SubmissionRequest_997	X12_Response_ConfirmReceiptReceived		
Retrieval of 997	N	X12_997_RetrievalRequest_00401	X12_004010_Response_997		
Implementation Acknowledgement Submission	N			X12_999_SubmissionRequest_005010X231A1	X12_Response_ConfirmReceiptReceived
Implementation Acknowledgement Retrieval	N			X12_999_RetrievalRequest_005010X231A1	X12_999_Response_005010X231A1
Application Reporting for Insurance	N			X12_824_Request_005010X186	X12_824_Response_005010X186

[48] Several PayloadType values referencing version 004010 of various administrative transactions are being retained since these implementation guides are not adopted under HIPAA. As such, the industry may continue using these implementation specifications to support their business needs and are not required to move to a 5010 version of them. The corresponding v5010 implementation guides have been added to the value set to support the industry's transition to v5010 in general.
[49] Ibid.
[50] Ibid.
[51] Ibid.
[52] Ibid.

Payloads to Support All ASC X12 Administrative Transactions for Batch					
Transaction Name	**HIPAA Mandated**	**Payload Type Value (Normative/Comprehensive)**			
		V4010A1[53]		**V5010**	
		Request	**Response**	**Request**	**Response**
Acknowledgements (Continued)					
General Acknowledgements Pick Up	N	X12_004010_Request_Acks	X12_004010_Response_Acks	X12_005010_Request_Acks	X12_005010_Response_Acks
No Acknowledgement File	N/A		X12_004010_Response_NoBatchAckFile		X12_005010_Response_NoBatchAckFile
Payload Receipt Confirmation	N/A	X12_Request_ConfirmReceipt	X12_Response_ConfirmReceiptReceived	X12_Request_ConfirmReceipt	X12_Response_ConfirmReceiptReceived
Errors					
CORE Envelope Error	N/A		CoreEnvelopeError		CoreEnvelopeError

[53] Several PayloadType values referencing version 004010 of various administrative transactions are being retained since these implementation guides are not adopted under HIPAA. As such, the industry may continue using these implementation specifications to support their business needs and are not required to move to a 5010 version of them. The corresponding v5010 implementation guides have been added to the value set to support the industry's transition to v5010 in general.

4.4.4 Enumeration Convention for PayloadType when Handling Non-X12 Payloads (Non-normative)

The Envelope metadata specification in §4.4.3 includes a PayloadType field that is enumerated for X12 payload types. This envelope may also be used to transport other types of payloads. In such cases, the convention for the *PayloadType* field is as follows:

<SDO>_<PayloadType>_<Version>_<Sub-version>

Note: SDO stands for Standards Development Organization.

For example, an NCPDP based NEWRX SCRIPT Standard Implementation Guide Version 8.1 transaction of NEWRX payload may specify *PayloadType* as *NCPDP_NEWRX_008_001*, an HL7 based ADT04 Version 2.3.1 payload may specify the *PayloadType* as *HL7_ADT04_2_3_1*.

5 CORE SAFE HARBOR

The Phase I CORE 153 Connectivity Rule version 1.1.0 provided a "Safe Harbor" that application vendors, providers, and health plans (or other information sources) could be assured would be supported by any CORE-certified trading partner. This Phase II Connectivity Rule extends the Safe Harbor by further specifying the connectivity that all CORE-certified organizations must implement and with which conformance must be demonstrated. As such, in the Phase I rule:

- **DOES NOT** require trading partners (e.g., a provider or a health plan) to discontinue using existing connections that do not match the rule.
- **DOES NOT** require that trading partners must use a CORE-complaint method for all new connections.
- **DOES NOT** require all CORE trading partners use only one method for any connections.
- **DOES NOT** require CORE-certified entity to do business with any trading partner or other CORE-certified entity.

CORE expects that in some circumstances, trading partners may agree to use different communication mechanism(s) and/or security requirements than those described in this rule to achieve the technical goals of the specific connection.

However, this Safe Harbor is the connectivity mechanism that a CORE-certified entity MUST use if requested by a trading partner. If the CORE-certified entity does not believe that this CORE Safe Harbor is the best connectivity mechanism for that particular trading partner, it may work with its trading partner to implement a different, mutually agreeable connectivity method. However, if the trading partner insists on using this Safe Harbor, the CORE-certified entity must accommodate that request. This clarification is not intended in any way to modify entities' obligations to exchange electronic transactions as specified by HIPAA or other federal and state regulations.

6 APPENDIX

6.1 Abbreviations and Definitions Used in this Rule

Term or Concept	Definition
ASC X12 Interchange	An ASC X12 Interchange is a graphic character string structured using delimited, tagged data concepts. An ASC X12 Interchange begins with an Interchange Control Header segment: Segment ID = ISA and ends with an Interchange Control Trailer segment: Segment ID – IEA. An ASC X12 Interchange may be composed of one or more Functional Groups (GS/GE Control Segments). An ASC X12 Functional Group is composed of one or more Transaction Sets (ST/SE Control Segments). An ASC X12 Interchange may be a Logical file or a physical file as determined by the originator of the Interchange. As such, a physical file may consist of one or more ASC X12 Interchanges. The ISA Interchange Control Header segment does not identify the content of any included Functional Groups. The Functional Group Control Header segment identifies the transaction set(s) in the Functional Group: GS08-480 Version/Release/Industry Indicator Code.
Asynchronous	A message exchange interaction is said to be asynchronous when the associated messages are chronologically and procedurally decoupled, e.g., in a request-response interaction, the client agent can process the response at some indeterminate point in the future when its existence is discovered. Mechanisms to do this include polling, notification by receipt of another message, etc. [WS Glossary, 2004]
Batch (Batch Mode, Batch Processing Mode)	Batch Mode is when the initial (first)[54] communications session is established and maintained open and active only for the time required to transfer a batch file of one or more transactions. A separate (second) communications session is later established and maintained open and active for the time required to acknowledge that the initial file was successfully received and/or to retrieve transaction responses. Batch Processing Mode[55] is also considered to be an asynchronous processing mode, whereby the associated messages are chronologically and procedurally decoupled. In a request-response interaction, the client agent can process the response at some indeterminate point in the future when its existence is discovered. Mechanisms to implement this capability may include: polling; notification by receipt of another message; or receipt of related responses (as when the request receiver "pushes" the corresponding responses back to the requestor), etc. Batch, (asynchronous) Processing Mode is from the perspective of the request initiator. If a Batch (asynchronous) request is sent via intermediaries, then such intermediaries may, or may not, use Batch Processing Mode to further process the request.
Batch Files (Payload)	A single submission of a message payload that contains one X12 Interchange containing one Functional Group containing one X12 transaction set consisting of more than one business transaction.

[54] CORE Phase I Glossary Definitions. http://www.caqh.org/pdf/COREPIGlossary.pdf
[55] CORE Phase I Glossary Definitions. http://www.caqh.org/pdf/COREPIGlossary.pdf

Term or Concept	Definition
Client	An entity that sends/relays a message to a Server.
CORE Safe Harbor	The connectivity requirements that application vendors, providers, and health plans (or other information sources) are required to support in order to provide assurance that these requirements are supported by any CORE-certified trading partners.[56]
Extensibility	Extensibility is a property of a system, format, or standard that allows evolution in performance or format within a common framework, while retaining partial or complete compatibility among systems that belong to the common framework.[57] Extensibility is a system design principle where the implementation takes into consideration future growth. It is a systematic measure of the ability to extend a system and the level of effort required to implement the extension. Extensions can be through the addition of new functionality or through modification of existing functionality. The central theme is to provide for change while minimizing the impact to existing system functions.[58]
Federal Information Processing Standards Security Requirements for Cryptographic Modules (FIPS 140-2)	The Federal Information Processing Standards Publication Series of the National Institute of Standards and Technology (NIST) is the official series of publications relating to standards and guidelines adopted and promulgated under the provisions of Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104-106) and the Computer Security Act of 1987 (Public Law 100-235). These mandates have given the Secretary of Commerce and NIST important responsibilities for improving the utilization and management of computer and related telecommunications systems in the Federal government. The NIST, through its Information Technology Laboratory, provides leadership, technical guidance, and coordination of government efforts in the development of standards and guidelines in these areas (http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf).
HTTP	Hypertext Transport Protocol Version 1.1 (IETF RFC 2616: http://www.ietf.org/rfc/rfc2616.txt).
HTTP MIME Multipart	MIME Multipart/related Content-type (IETF RFC 2388: http://www.ietf.org/rfc/rfc2388.txt).
Interoperability	Interoperability is the capability of different information technology systems, software applications and networks to communicate, execute programs, exchange data accurately, effectively and consistently, among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units and to use the information that has been exchanged.[59] Interoperability also requires no specific architecture and is independent of vendors and their various operating systems, programming languages, hardware, and network infrastructure**.** Functional interoperability is the capability to reliably exchange information without errors. Semantic interoperability allows systems to interpret and make effective use of the information exchanged among systems[60].

[56] CORE Phase I Rule 153: Eligibility and Benefits Connectivity Rule Version 1.0.0
[57] http://www.atis.org/glossary/definition.aspx?id=7853 ATIS (Alliance for Telecommunications Industry Solutions http://www.atis.org/about/index.asp
[58] http://en.wikipedia.org/wiki/Extensibility
[59] Adapted from http://engineers.ihs.com/document/abstract/AQSBFBAAAAAAAAAA ANSI Information Technology – Vocabulary – Part 1: Fundamental Terms
[60] HIMSS Position Statement: Adoption of HITSP Interoperability Specifications July 2007

Term or Concept	Definition
Interoperability Specification[61]	An Interoperability Specification focuses on a set of constrained standards for information interchange that address the core requirements of the Use Cases. It does not define all functions, constructs and standards necessary to implement a conforming system in the real world environment. An Interoperability Specification defines how two or more systems exchange standard data content in a standard manner. Interoperability Specifications define the necessary business and technical actors, the transactions between them including the message, content and terminology standards for the actual information exchange. Interoperability Specifications do not specify the functional requirements or behaviors of the systems or applications. Interoperability Specifications, unless otherwise noted, are not intended to define or prescribe any system architecture or implementation. At the most basic level, the Interoperability Specifications define specific information exchange standards that are to be used by any two systems. Information exchange must be placed within the context of a transaction between defined technical actors which fulfill higher level business requirements derived from the use cases. In some cases the necessary technical actors may require some architectural structure or make some assumptions involving synchronous or asynchronous data exchanges, or require specific type of exchange, such as a message or document. These requirements may constrain to some degree the total range of choices regarding system architectures. When constraints are necessary to meet the use case requirements, the Interoperability Specification will note this and will retain as much architectural neutrality as possible. When appropriate, Interoperability Specifications may provide architectural examples and discuss considerations of such examples. HITSP and ONC do not define "Interoperability," but, do define "Interoperability Specification."
Large Batch Files (Payload)	A single submission of a message payload that contains more than one ASC X12 Interchange, each of which may contain one or more Functional Groups, each of which may contain one or more ASC X12 transaction sets.
Large Volume of Single Real time Transactions (Synchronous)	A high number of Real time transactions arriving at the receiving system concurrently. CORE defines large volume as "X"% of an organization's average daily received transaction volume (based on all trading partners) within one minute. "X" is defined by organization.
Message Encapsulation Layer	This refers to the Open Systems Interconnect (OSI) layers 5 and 6.
Message Envelope Standard A	HTTP MIME Multipart, described in Section "Specifications for HTTP MIME Multipart".
Message Envelope Standard B	SOAP+WSDL, described in Section "Specifications for SOAP + WSDL".
MIME	Multipurpose Internet Message Extensions (IETF RFCs 2045 to RFC 2049) [http://www.ietf.org/rfc/rfc2045.txt].
MTOM	W3C Message Transmission Optimization Mechanism (http://www.w3.org/TR/soap12-mtom/).
Normative	In standards terminology, "normative" means "considered to be a prescriptive part of the standard" [Wikipedia].

[61] HITSP Interoperability Specification: EHR Lab Terminology Component HITSP/ISC-35 October 20, 2006 Version 1.2

Term or Concept	Definition
Non-normative	Informational, not intended to be part of the specification.
OSI	Open Systems Interconnection Basic Reference Model (OSI Reference Model, or OSI Model for short) is a layered, abstract description for communications and computer network protocol design. From top to bottom, the OSI Model consists of the Application, Presentation, Session, Transport, Network, Data Link and Physical Layers [Wikipedia].
Open Standard [62]	"Open Standards" are those standards made available to the general public and are developed (or approved) and maintained via a collaborative and consensus driven process. Open Standards facilitate interoperability and data exchange among different products or services and are intended for widespread adoption.
Payload	The essential data that is being carried within a packet or other transmission unit. The payload does not include the "overhead" data required to get the packet to its destination. A packet is the unit of data that is routed between an origin and a destination on the Internet or any other packet-switched network. When any file (e-mail message, HTML file, Graphics Interchange Format file, Uniform Resource Locator request, and so forth) is sent from one place to another on the Internet, the Transmission Control Protocol (TCP) layer of TCP/IP divides the file into "chunks" of an efficient size for routing. Each of these packets is separately numbered and includes the Internet address of the destination. The individual packets for a given file may travel different routes through the Internet. When they have all arrived, they are reassembled into the original file (by the TCP layer at the receiving end).[63]
Performance	According to CORE's Phase I Connectivity Rule, performance is defined in only two components: Response time – the time required to receive an Eligibility Request, process it completely and send an appropriate response, as specified in CORE's Phase I Eligibility and Benefits Rules and Policies for Real time [64] and Batch [65] exchanges. System Availability – the time an information source's (health plan, clearinghouse/switch or other intermediary system) processing system is capable of properly processing Eligibility Request/Response transactions, as specified in CORE's Phase I Eligibility and Benefits Rules and Policies for system availability[66].
Performance Evaluation Criteria	For the purpose of evaluating the measurable performance dimensions of potential messaging methodologies to be used in Real time healthcare transactions, Performance Evaluation Criteria may include: Response Time – the time required to receive an Eligibility Request, process it completely, and send an appropriate response.[67] Maximum Arrival Rate Before Saturation – the maximum number of properly formed arriving Eligibility Request transactions per time period (usually seconds or minutes), above which the ability for increased acceptance for further processing stops.[68] Overhead Information – Digital information transferred across the functional interface between a user and a telecommunications system, or between functional units within a telecommunications system, for the purpose of directing or

[62] International Telecommunication Union – Open Standards Definition. http://www.itu.int/ITU-T/othergroups/ipr-adhoc/openstandards.html
[63] SearchSecurity.com. http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci214475,00.html
[64] CORE Phase I 156: Eligibility and Benefits Real Time Response Time Rule
[65] CORE Phase I 155: Eligibility and Benefits Batch Response Time Rule
[66] CORE Phase I 157: Eligibility and Benefits System Availability
[67] CORE Phase I 156: Eligibility and Benefits Real Time Response Time Rule; and CORE Phase I 155: Eligibility and Benefits Batch Response Time Rule
[68] http://www.cs.washington.edu/homes/lazowska/qsp/Contents.pdf Quantitative System Performance, Chapter 5.2.1. Transaction Workloads (Page 72)

Term or Concept	Definition
	controlling the transfer of user information or the detection and correction of errors. Note: Overhead information originated by the user is not considered to be system overhead information. Overhead information generated within the communications system and not delivered to the user is system overhead information. Thus, the user throughput is reduced by both overheads while system throughput is reduced only by system overhead.[69] Capacity – the maximum number of completed Eligibility Request/ Response transaction sets per specific time period. Quality of Service – the number of properly and accurately completed Eligibility Request/Response transaction sets divided by the number of properly submitted transactions (Requests). When making such performance measurements and evaluations, it is important to consider the architecture of networks and systems to assure their similarity, and/or to assess the relevance and impact of any differences.
Real time (Real time Mode, Real time Processing Mode) [70]	Real time Mode [71] is when an entity is required to immediately send a single transaction and receive a single, related response within a single communications session, which is established and maintained open and active until the required response is received by the entity initiating that session. Communication is complete when the session is closed. Real time Mode & Real time Processing Mode are also considered to be a synchronous processing mode. (See Synchronous). Real time, or synchronous, Processing Mode is from the perspective of the request initiator.
Safe Harbor	A "Safe Harbor" is generally defined as a statutory or regulatory provision that provides protection from a penalty or liability.[72] In many IT-related initiatives, a safe harbor describes a set of standards/guidelines that allow for an "adequate" level of assurance when business partners are transacting business electronically.
Secure Sockets Layer (SSL)	See Transport Layer Security.
Server	An entity that receives a message from a Client, which it may process, or relay to another Server.
SOAP	W3C Simple Object Access Protocol Version 1.2. (http://www.w3.org/TR/soap12-part1/)
Standard	A standard is a document, established by consensus and approved by a recognized body that provides, for common and repeated use, rules, guidelines or characteristics for activities or their results, aimed at the achievement of the optimum degree of order in a given context.[73]
Standard Development Organization	Standards Development Organizations (SDOs) are organizations whose processes are accredited by ANSI. A SDO may also include non-ANSI accredited organizations such as W3C, OASIS, ISO, UN/CEFACT and IETF.
Support [Supported]	Means that the entity must have the capability as specified and required.

[69] http://www.atis.org/tg2k/ and search "Overhead Information" ATIS (Alliance for Telecommunications Industry Solutions http://www.atis.org/about.shtml
[70] CORE Phase I Glossary Definitions. www.caqh.org/pdf/COREPIGlossary.pdf
[71] CORE Phase I Glossary Definitions. http://www.caqh.org/pdf/COREPIGlossary.pdf
[72] Merriam-Webster's Dictionary of Law. Merriam-Webster, Inc., 28 May, 2007. <Dictionary.com http://dictionary.reference.com/browse/safeharbor>
[73] http://isotc.iso.org/livelink/livelink/fetch/2000/2122/830949/3934883/3935096/07_gen_info/faq.html

Term or Concept	Definition
Submitter Authentication Standard C	Username-Password, described in Sub-section "Submitter Authentication Handling."
Submitter Authentication Standard D	X.509 Certificate based Authentication over SSL[74], described in Sub-section "Submitter Authentication Handling."
Synchronous	The application sending the request message waits for the response, which is returned on the same communications connection (i.e., synchronous request/reply). This message exchange pattern is used for most real time transactions.
Transport Layer Security (TLS)	Transport Layer Security (TLS)[75] and its predecessor, Secure Sockets Layer (SSL), are cryptographic protocols that "provide communications security over the Internet". TLS and SSL encrypt the segments of network connections above the Transport Layer, using symmetric cryptography for privacy and a keyed message authentication code for message reliability. Several versions of the protocols are in widespread use in applications like web browsing, electronic mail, Internet faxing, instant messaging and voice-over-IP (VoIP). TLS is an IETF standards track protocol, last updated in RFC 5246, and is based on the earlier SSL specifications developed by Netscape Corporation (http://tools.ietf.org/html/rfc5246).
WSDL	W3C Web Services Definition Language Version 1.1 (http://www.w3.org/TR/2001/NOTE-wsdl-20010315).

[74] Reference §3.1 for more information regarding the use of SSL and the optional use of TLS 1.0.
[75] http://en.wikipedia.org/wiki/Transport_Layer_Security

6.2 References

Note: These were used for Rule creation as well as to create the analysis artifacts as part of CORE Phase II Connectivity.

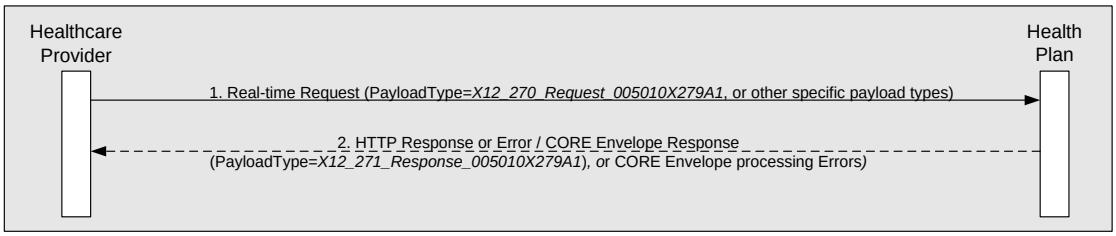
Author	Document Name	Location
CORE	Claim Status Rule Test Scenario	CORE Operating Rule 250
HL7 (Health Level 7)	HL7 Object Identifier (OID) Registry	http://www.hl7.org/oid/index.cfm
Internet Assigned Numbers Authority (IANA)	IANA Private Enterprise Number (PEN) aka "OID" Registration Page	http://www.iana.org/cgi-bin/enterprise.pl
Internet Engineering Task Force (IETF)	Key Words for use in RFCs to Indicate Requirement Levels	http://www.ietf.org/rfc/rfc2119.txt
Internet Engineering Task Force (IETF)	Uniform Resource Identifier (URI): Generic Syntax	http://www.gbiv.com/protocols/uri/rfc/rfc3986.html
Internet Engineering Task Force (IETF)	Hypertext Transfer Protocol – HTTP 1.1	http://tools.ietf.org/html/rfc2616.txt
Internet Engineering Task Force (IETF)	HTTP Authentication: Basic and Digest Access Authentication	http://tools.ietf.org/html/rfc2617.txt
Internet Engineering Task Force (IETF)	The MIME Multipart/Form-Data (RFC 2388)	http://www.ietf.org/rfc/rfc2388.txt
Internet Engineering Task Force (IETF)	TLS 1.1 Specification	http://tools.ietf.org/html/rfc4346.txt
Internet Engineering Task Force (IETF)	Universally Unique IDentifier (UUID) URN Namespace	ftp://ftp.rfc-editor.org/in-notes/rfc4122.txt
OASIS	Web Services Reliable Messaging Protocol 1.1 (WS-RM)	http://docs.oasis-open.org/ws-rx/wsrm/v1.1/wsrm.html
OASIS	Web Service Security Core Specification 1.1	http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf
OASIS	Web Service Security SOAP Message Security 1.1	http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-errata-os-SOAPMessageSecurity.pdf
OASIS	Web Service Secure Conversation 1.3	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html
OASIS	Universal Description, Discovery and Integration (UDDI) 1.0	http://www.oasis-open.org/committees/uddi-spec/doc/contribs.htm#uddiv1
OASIS	ebXML Message Service Specification v2.0	http://www.oasis-open.org/committees/ebxml-msg/documents/ebMS_v2_0.pdf
W3C (World Wide Web Consortium)	Extensible Mark-up Language (XML) 1.0 (Fourth Edition)	http://www.w3.org/TR/2006/REC-xml-20060816/
W3C (World Wide Web Consortium)	Namespaces in XML 1.0 (Second Edition)	http://www.w3.org/TR/2006/REC-xml-names-20060816
W3C (World Wide Web Consortium)	Canonical XML Version 1.0	http://www.w3.org/TR/2002/REC-xmldsig-core-20020212
W3C (World Wide Web Consortium)	XML Schema Part 1: Structures Second Edition	http://www.w3.org/TR/2004/REC-xmlschema-1-20041028
W3C (World Wide Web Consortium)	XML Schema Part 2: Datatypes Second Edition	http://www.w3.org/TR/2004/REC-xmlschema-2-20041028
W3C (World Wide Web Consortium)	XML Signature Syntax and Processing	http://www.w3.org/TR/2002/REC-xmldsig-core-20020212
W3C (World Wide Web Consortium)	XML Encryption Syntax and processing	http://www.w3.org/TR/2002/REC-xmlenc-core-20021210
W3C (World Wide Web Consortium)	Simple Object Access Protocol (SOAP) 1.2	http://www.w3.org/TR/soap12-part1/
W3C (World Wide Web Consortium)	SOAP Message Transmission Optimization Mechanism (MTOM)	http://www.w3.org/TR/2005/REC-soap12-mtom-20050125
W3C (World Wide Web Consortium)	Web Services Description Language (WSDL) 1.1	http://www.w3.org/TR/2001/NOTE-wsdl-20010315
Web Services Interoperability Organization	SOAP Basic Profile 1.1	http://www.ws-i.org/Profiles/BasicProfile-1.1-2006-04-10.html

6.3 Sequence Diagrams

The UML sequence diagrams below show interactions between a client (typically, a Health Care Provider) and a server (typically, a Health Plan). When the interactions include multiple requests/responses, each pair of requests and its corresponding (synchronous) response is shown encapsulated in a white rectangle. Each request followed by synchronous response (shown in a single white rectangle) is in a client-server connection that can be expected to be opened for a request and closed after the corresponding synchronous response is received. Subsequent requests/responses occur in new client-server connections. Servers are stateless and are not assumed to keep session information between connections, unless such information is sent as part of the requests (e.g., using 999 or TA1 payloads).

6.3.1 Real Time Interaction

The UML sequence diagram below shows a typical Real time Interaction between a Healthcare Provider and a Health Plan. The interactions are described in the diagram below.



The following describes the typical Real time interaction as shown in the above diagram.

Message Sequence	Description	Reference to Section 4.4.3.1 Payload Type Table Transaction Name Column
1	Healthcare Provider submits a Real time request to the Health Plan, using payload type as X12_270_Request_005010X279A1, or one of the specific payload types (shown in section 4.4.3.1).	Health Care Eligibility Benefit Inquiry and Response
2	Health Plan responds (synchronously to request message 1) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (Payload type is X12_271_Response_005010X279A1, or error).	Health Care Eligibility Benefit Inquiry and Response

6.3.2 Batch Interaction

These sequence diagrams are non-normative examples of two types of Batch interactions. The Batch interactions can be conducted using specific payload types as shown in 6.3.2.1 or with Mixed Payload types as show in 6.3.2.2.

6.3.2.1 Batch Interaction for Specific Payload Types

The UML sequence diagram below shows a typical Batch Interaction between a Healthcare Provider and a Health Plan specifically for ASC X12 270/271 batch payloads (the same interaction applies to ASC X12 276/277 batch payloads).

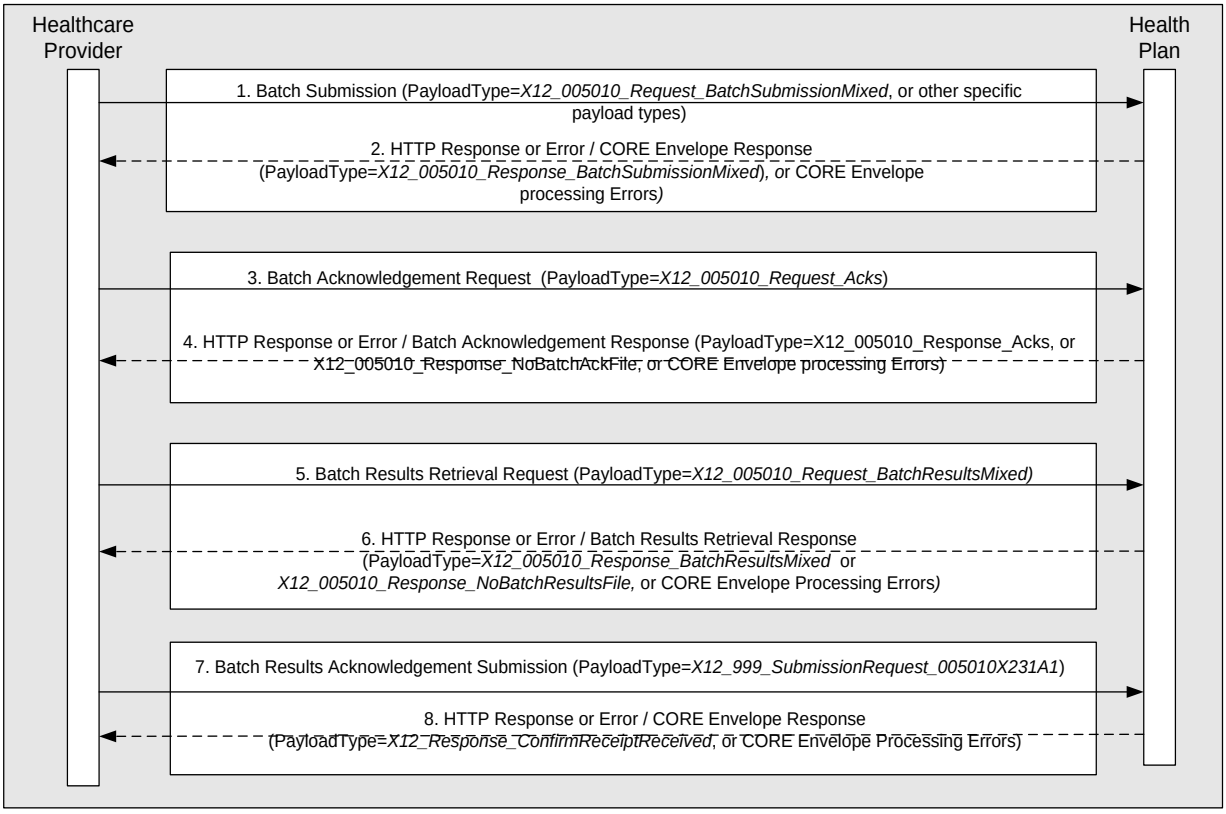
The following describes the Batch interaction as shown in the above diagram.

Message Sequence	Description	Reference to Section 4.4.3.2 Payload Type Table Transaction Name Column
1	Healthcare Provider submits a Batch of requests to the Health Plan, using payload type as X12_270_Request_005010X279A1.	Health Care Eligibility Benefit Inquiry and Response
2	Health Plan responds (synchronously to request message 1) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), indicating that the Batch was received (e.g., payload type = X12_BatchReceiptConfirmation) and the CORE envelope was processed (with or without errors).	Health Care Eligibility Benefit Inquiry and Response

Message Sequence	Description	Reference to Section 4.4.3.2 Payload Type Table Transaction Name Column
3	Healthcare Provider sends a Request to the Health Plan to solicit the acknowledgement (X12_999_RetrievalRequest_005010X231A1) for the Batch file that was just submitted.	Implementation Acknowledgement Retrieval
4 I	Health Plan responds (synchronously to request message 3) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), with the X12_999_Response_005010X231A1 or an X12_TA1_Response_005010X231A1 acknowledgement. If no v5010 999 or TA1 is ready for pickup, Health Plan sends a CORE Envelope with payload type set to X12_005010_Response_NoBatchAckFile.	Implementation Acknowledgement Retrieval
5	Healthcare Provider sends a Request to the Health Plan to solicit the Results for the Batch file that was submitted in message sequence 1 using X12_005010_Request_Batch_Results_271.	Health Care Eligibility Benefit Inquiry and Response
6	Health Plan responds (synchronously to request message 5) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), with the payload type set to X12_271_Response_005010X279A1, and sends the result file as payload.	Health Care Eligibility Benefit Inquiry and Response
7	Healthcare Provider submits the acknowledgement (payload type X12_999_SubmissionRequest_005010X231A1, X12_TA1_SubmissionRequest_00501X231A1) to the Health Plan. This acknowledgment submission is required by CORE Phase I and Phase II Rules.	Implementation Acknowledgement Submission (Request)
8	Health Plan responds (synchronously to request message 7) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), indicating that the Batch results acknowledgement was received (payload type = X12_Response_ConfirmReceiptReceived) and the CORE envelope was processed (with or without errors).	Implementation Acknowledgement Submission (Response)

6.3.2.2 Batch Interaction for Mixed Payload Types

The UML sequence diagram below shows a Mixed Payload Type Batch Interaction between a Healthcare Provider and a Health Plan.



The following describes the typical Mixed Batch interaction as shown in the above diagram.

Message Sequence	Description	Reference to Section 4.4.3.2 Payload Type Table Transaction Name Column
1	Healthcare Provider submits a Batch of requests to the Health Plan, using payload type as BatchSubmissionMixed (e.g., payload type=X12_005010_Request_BatchSubmissionMixed), or one of the specific payload types (shown in section 4.4.3.2).	Batch Submission (mixed payload types)
2	Health Plan responds (synchronously to request message 1) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), indicating that the Batch was received (e.g., payload type = X12_005010_Response_BatchSubmissionMixed) and the CORE envelope was processed (with or without errors).	Batch Submission (mixed payload types)
3	Healthcare Provider sends a Request to the Health Plan to solicit the acknowledgement (ASC X12 v5010 999 or TA1 for the Batch file that was	General Acknowledgements

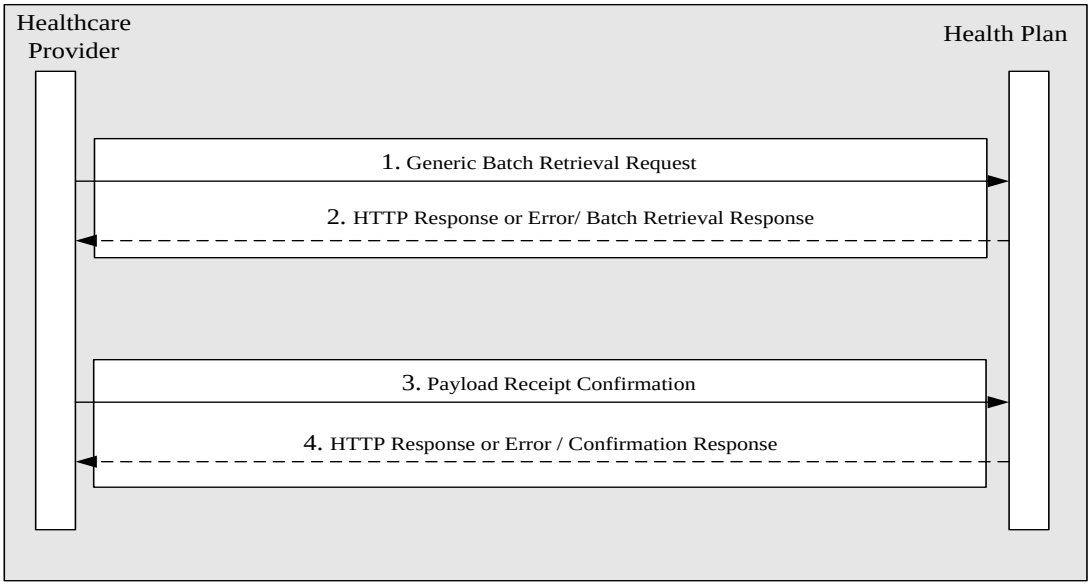
Message Sequence	Description	Reference to Section 4.4.3.2 Payload Type Table Transaction Name Column
	just submitted.	Pick Up
4	Health Plan responds (synchronously to request message 3) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), with the ASC X12 v5010 999 or an X12_TA1 acknowledgement. If no v5010 999 or TA1 is ready for pickup, Health Plan sends a CORE Envelope with payload type set to X12_005010_Response_NoBatchAckFile, or X12_005010_Response_Acks.	No Acknowledgement File or General Acknowledgements Pickup
5	Healthcare Provider sends a Request to the Health Plan to solicit the Results for the Batch file that was submitted in message sequence 1.	Batch Results Retrieval (mixed payload types)
6	Health Plan responds (synchronously to request message 5) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), with the payload type set to X12_005010_Response_BatchResultsMixed, and sends the result file as payload. If no results file is ready for pickup, Health Plan sends a CORE Envelope with payload type set to X12_005010_Response_NoBatchResultsFile.	• Batch Results Retrieval (mixed payload types) • No Acknowledgement File
7	Healthcare Provider submits the acknowledgement (payload type X12_999_SubmissionRequest_005010X231A1 or X12_TA1_SubmissionRequest_005010X231A1) to the Health Plan. This acknowledgment submission is required by CORE Phase I and Phase II Rules.	Implementation Acknowledgement Submission
8	Health Plan responds (synchronously to request message 7) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), indicating that the Batch results acknowledgement was received (payload type = X12_Response_ConfirmReceiptReceived) and the CORE envelope was processed (with or without errors).	Implementation Acknowledgement Submission

6.3.3 Generic Batch Retrieval Request and Receipt Confirmation

The UML sequence diagram below shows a Generic Batch Retrieval Request and Receipt Confirmation interaction between a Healthcare Provider and a Health Plan.

If this interaction is used for a Healthcare Claim Acknowledgement, there is a pre-condition in this interaction. In this case, the Healthcare Provider must have previously submitted a claim to the Health Plan and the Healthcare Provider is now requesting information about that claim.

The message interactions are described in the diagram below.



The following describes the typical Batch Retrieval interaction as shown in the above diagram.

Message Sequence	Description	Reference to Section 4.4.3.2 Payload Type Table Transaction Name Column
1	Healthcare Provider submits a Generic Batch Retrieval Request to the Health Plan, using a valid PayloadType from the table in section 4.4.3.2.	• Health Care Claim Acknowledgement • Health Care Claim Payment/Advice • Health Care Claim Pending Status Information
2	Health Plan responds synchronously in Real time with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), indicating that the Batch was received and the CORE envelope was processed without errors	• Health Care Claim Acknowledgement • Health Care Claim Payment/Advice • Health Care Claim Pending Status Information
3[76]	Health Care Provider sends to Health Plan a confirmation that	Payload Receipt Confirmation

[76] Since step 3 is a new connection (session) from the Provider to the Plan, for there to be an association of the Receipt Confirmation sent in this connection with the previously received retrieval response, this confirmation needs to be a 999 or a TA1.

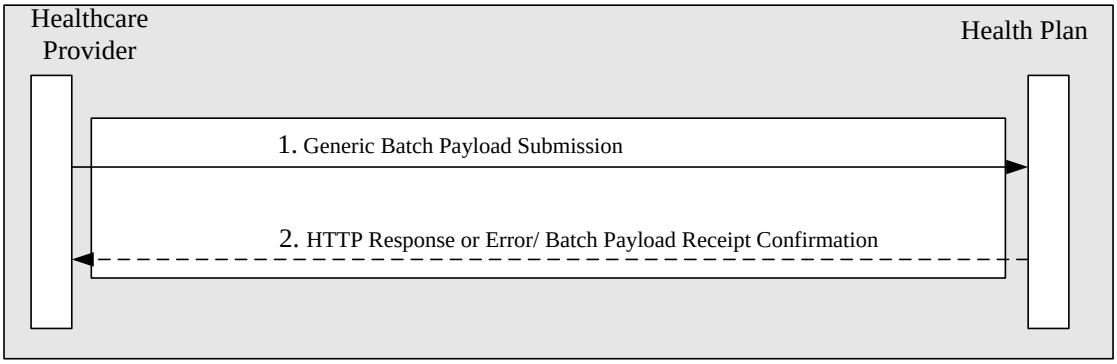
Message Sequence	Description	Reference to Section 4.4.3.2 Payload Type Table Transaction Name Column
	it received the payload sent in step 2.	
4	Health Plan responds (synchronously to request message 3) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response, or CORE Envelope Error.	Payload Receipt Confirmation

6.3.4 Generic Batch Submission with Batch Payload and Synchronous Payload Receipt Confirmation

The UML sequence diagram below shows a Generic Batch Payload Submission and Synchronous Payload Receipt Confirmation interaction between a Healthcare Provider and a Health Plan (Note: The submission of a Generic Batch may be performed by a stakeholder other than a Healthcare Provider, e.g., a Health Plan Sponsor.)

This interaction allows us to specify only the steps necessary for a Client (typically, a Healthcare Provider) to perform a Batch Payload submission to a Server (typically, a Health Plan) and receive the corresponding synchronous receipt confirmation. In terms of the actual request and response, the Generic Batch Submission Interaction also happens to be a subset of entire Batch Interaction (§6.3.2). The part of the WSDL (for SOAP envelope) that has been extended to describe the Generic Batch Submission Interaction shows how that interaction re-uses the existing Batch Submission request/response messages, for which examples are provided in section 4. For this reason, footnotes have been provided where the existing Batch Submission examples also serve as examples for the Generic Batch Submission message interaction.

The message interactions are described in the diagram below.



The following describes the typical Generic Batch Submission interaction as shown in the above diagram.

Message Sequence	Description	Reference to Section 4.4.3.2 Payload Type Table Transaction Name Column
1	Healthcare Provider submits a Batch Payload to a Health Plan using a valid PayloadType from the table in §4.4.3.	• Health Care Claim Request for Additional Information • Request for Information in Support of a Disability Claim • Health Care Claim: Institutional • Health Care Claim: Professional • Health Care Claim: Dental

Message Sequence	Description	Reference to Section 4.4.3.2 Payload Type Table Transaction Name Column
		• Payroll Deducted and Other Group Premium Payment for Insurance Products • Benefit Enrollment and Maintenance • Health Care Benefit Coordination Verification • Health Care Predetermination – Professional • Health Care Predetermination – Institutional • Doctors First Report of Injury • Health Care Service: Data Reporting
2	Health Plan responds synchronously in Real time with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), indicating that the Batch was received and the CORE envelope was processed without errors.	• Health Care Claim: Institutional • Health Care Claim: Professional • Health Care Claim: Dental • Payroll Deducted and Other Group Premium Payment for Insurance Products • Benefit Enrollment and Maintenance • Health Care Benefit Coordination Verification • Health Care Predetermination – Professional • Health Care Predetermination – Institutional • Doctors First Report of Injury • Health Care Service: Data Reporting

