Phase IV CAQH CORE 470 Connectivity Rule Version 4.0.0

Draft for Technical Work Group Ballot

April 2015

DOCUMENT CHANGE HISTORY

Description of Change	Name of Author	Date Published
Full draft rule for Straw poll with all sections and the following modifications based on Subgroup discussions	CAQH CORE Staff	10/09/14
Non-substantive and clarifying adjustments to address CSSG straw poll results • footnote inserted in Section 1.1 defining the term agent • inserted text in Section 3.2 to enable the optional use of TLS 1.1 or higher in lieu of SSL 3.0 • added text in Section 5 indicating that list of recommendations is not intended to be exhaustive or prohibitive • added to Section 7, Real Time transaction message interaction sequence diagrams for ASC X12N 820, ASC X12N 834 and ASC X12N 837 transactions.	CAQH CORE Staff	January 2015
Review and disposition of Technical Work Group straw poll results and comments for Technical Work Group Ballot • Added non-substantive clarifying text to Section 3.4 • Non-substantive adjustment to Section 4.1 text for clarification • Non-substantive adjustment to Appendix 7.2 Real Time definition • Non-substantive adjustments to Appendix 7.3 Sequence Diagrams reformatted for improved readability	Technical Work Group	March 2015

Table of Contents

1 BACKGROUND

Each Phase of CAQH CORE Operating Rules builds on the previous phases to encourage feasible industry progress. Continuing to build on the Phase I, II, & III CAQH CORE Operating Rules, the Affordable Care Act Section 1104 has mandated that CAQH CORE Operating Rules should be adopted that include rules around the health care claims and encounter reporting, health care services request for review and response, health plan premium payment, benefit enrollment and maintenance transactions, and attachments to allow the industry to leverage its investment in the Phase I, II, and III CAQH CORE Rules and apply them to exchanging the following HIPAA mandated transactions:

- ASC X12N 005010X223 Health Care Claim Institutional (837) ASC X12N 005010X222 Health Care Claim Professional (837) and ASC X12N 005010X224 Health Care Claim Dental (837) and their respective errata (collectively hereafter referenced as ASC X12N v5010 Claim)
- ASC X12N 005010X217 Health Care Services Review – Request for Review and Response (278) and associated errata (hereafter referenced as ASC X12N v5010 278 Request and Response and referred to as prior authorization in general)
- ASC X12N 005010X218 Payroll Deducted and Other Group Premium Payment for Insurance Products (820) and associated errata (hereafter referenced as ASC X12N v5010 820)
- ASC X12N 005010X220 Benefit Enrollment and Maintenance (834) and associated errata (hereafter referenced as ASC X12N v5010 834)

The use of the ASC X12N v5010 820 and ASC X12N v5010 834 transactions by the Insurance Exchanges[1] is out of scope for this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0.

Note: HHS has not adopted a standard for health claims attachments or indicated what standard(s) it might consider for the transaction, and an effective date for these operating rules is not included in the ACA. Thus, the immediate focus of this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 will not include attachments.

The Phase IV CORE 470 Connectivity Rule v4.0.0 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. Therefore, a key goal of this Phase IV CAQH CORE 470 Connectivity Operating Rule v4.0.0 is to continue to facilitate the industry's momentum to increase access to the HIPAA-mandated administrative transactions and to enable all HIPAA covered entities or their agents[2], business associates, intermediaries, and vendors to build on and extend the connectivity and infrastructure capabilities established for the eligibility and claim status transactions in Phase I and II of CAQH CORE operating rules, which were then applied to the electronic remittance advice transaction in Phase III of CAQH CORE operating rules.

An important component of this goal is to further facilitate interoperability by moving the healthcare industry to a single message envelope[3] standard along with a single submitter authentication[4] method as set forth in Section 2.2.2 of the ACA mandated Phase II CAQH CORE Connectivity 270 Rule v2.2.0.

---

[1] 45 CFR §155.20 Definitions. *Exchange* means a governmental agency or non-profit entity that meets the applicable standards of this part and makes QHPs available to qualified individuals and/or qualified employers. Unless otherwise identified, this term includes an Exchange serving the individual mark for qualified individuals and a SHOP serving the small group market for qualified employers, regardless of whether the Exchange is established and operated by a State (including a regional Exchange or subsidiary Exchange) or by HHS.

[2] One who agrees and is authorized to act on behalf of another, a principal, to legally bind an individual in particular business transactions with third parties pursuant to an agency relationship. Source: West's Encyclopedia of American Law, edition 2. Copyright 2008 The Gale Group, Inc. All rights reserved.

[3] See §7.2 Abbreviations and Definitions Used in this Rule

[4] Ibid.

An ancillary goal of this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 is to reinforce and clarify the "safe harbor"[5] established in Phase I CAQH CORE 153 Connectivity Rule and Phase II CAQH CORE 270 Connectivity Rule that application vendors, providers and health plans, business associates or other intermediaries can be assured will be supported by any HIPAA covered entity or its agent. Essentially, all HIPAA covered entities or their agents must support the connectivity requirements as specified in this rule. Clarification of the "safe harbor" addresses the requirement that when a HIPAA covered entity or its agent are exchanging the transactions addressed by this rule using any other connectivity method as permitted by the CAQH CORE Safe Harbor the Processing Mode requirements specified in the Phase IV CAQH CORE-required Processing Mode Table also apply. (See §5.) However, 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 trading partners must use this method for all new connections. CAQH 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 Affordable Care Act Mandates

This CAQH CORE Rule is part of a set of rules that addresses requirements in Section 1104 of the Affordable Care Act (ACA). Section 1104 contains an industry mandate for the use of operating rules to support implementation of the HIPAA standards. Using successful, yet voluntary, national industry efforts as a guide, Section 1104 defines operating rules as "the necessary business rules and guidelines for the electronic exchange of information that are not defined by a standard or its implementation specifications." As such, operating rules build upon existing healthcare transaction standards. The ACA outlines three sets of healthcare industry operating rules to be approved by the Department of Health and Human Services (HHS) and then implemented by the industry.

The third set of ACA-mandated operating rules address the health care claims or equivalent encounter information transactions, enrollment and disenrollment in a health plan, health plan premium payments, claims attachments, and referral certification and authorization.[6] The ACA requires HHS to adopt a set of operating rules for these five transactions by July 2014[7]. In a letter dated 09/12/12 to the Chairperson of the National Committee on Vital and Health Statistics (NCVHS),[8] the Secretary of HHS designated CAQH CORE as the operating rule authoring entity for the remaining five HIPAA-mandated electronic transactions.

Section 1104 of the ACA also adds the health claims attachment transaction to the list of electronic healthcare transactions for which the HHS Secretary must adopt a standard under HIPAA. The ACA requires the health claims attachment transaction standard to be adopted by 01/01/14, in a manner ensuring that it is effective by 01/01/16[9].

1.2 Industry Neutral Standards Addressed in this Rule

This Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 addresses industry neutral transport, transport security, message envelope, and submitter authentication standards as well as CAQH CORE specified message envelope metadata,[10, 11] for both real time and batch processing of transmitted transactions, and communications-level errors and acknowledgements. These standards include the public Internet, Hypertext Transport Protocol (HTTP), Secure Sockets Layer (SSL), Transport Layer Security (TLS), SOAP, MTOM, XSD, WSDL, and the X.509 Digital Certificate for submitter authentication.

---

[5] See §5 Safe Harbor and §7.2 Abbreviations and Definitions Used in this Rule

[6] The first set of operating rules under ACA Section 1104 applies to eligibility and claim status transactions; these operating rules were effective 01/01/13. The second set of operating rules applies to EFT and ERA; these operating rules were effective 01/01/14.

[7] This date is statutory language and statutory language can be changed only by Congress.

[8] 09/12/12 HHS Letter from the Secretary to the Chairperson of NCVHS.

[9] This date is statutory language and statutory language can be changed only by Congress.

[10] See §7.2 Abbreviations and Definitions Used in this Rule

[11] See §4.4 Envelope Metadata Fields, Descriptions, Intended Use and Syntax/Value Sets and §7.2 Abbreviations and Definitions Used in this Rule

2 ISSUES TO BE ADDRESSED AND BUSINESS JUSTIFICATION

2.1 Problem Space

Recognizing that the healthcare industry uses multiple connectivity methods for electronic administrative transactions – some based on open standards, others on proprietary approaches – in Phases I, II and III the CAQH Committee on Operating Rules for Information Exchange (CORE®) aimed to fill that gap by formulating connectivity and security rules to support healthcare industry specific transactions. Requirements related to connectivity, infrastructure, e.g., response times, companion guides, system availability, etc., were addressed in multiple transaction-specific operating rules. The Phase I and Phase II CAQH CORE Connectivity Operating Rules specifically addressed the message envelope, corresponding envelope metadata, vocabularies and semantics needed, real time and batch payload processing modes, and the industry's developing use of the public Internet. However, there were challenges experienced by the industry when implementing the Phase I and Phase II CAQH CORE Connectivity Operating Rules, which this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 addresses, e.g.:

- Complexity: provides a simpler and more prescriptive rule with fewer options (e.g., single envelope standard, and single authentication standard)
- Transaction Support: provides more robust and uniform support for handling transaction payload by requiring MTOM for SOAP (both Real Time and Batch mode); provides better support for the new set of transactions relative to the previous rules, e.g., by supporting additional message interactions
- Security: improves security by removing Username+Password which is a weak form of B2B authentication, and by requiring the use of only X.509 Client Certificate based authentication over SSL/TLS which is a stronger form of authentication. Improved support for FIPS 140-2 compliance for entities requiring such compliance, in terms of transport security and message envelope security

2.2 CAQH CORE Process in Addressing the Problem Space

As part of the development of the Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 environmental scans as well as extensive business and market analysis were conducted to gain insights into the current industry landscape regarding legislative, market movements and national initiatives. The results of these efforts identified several potential opportunity areas as a focus for the Phase IV CAQH CORE 470 Connectivity Operating Rule v4.0.0. Table 2.2-1 below summarizes at a high level the potential opportunity areas identified.

Table 2.2-1 Potential Phase IV Connectivity Rule Opportunities
Opportunity Area A: Improving Rule Language/Clarity
Rule Opportunity #1: Improve clarity around Real time and Batch requirements, error handling
Rule Opportunity #2: Address CAQH CORE Connectivity Rule v2.2.0 implementer feedback specific to technical issues
Opportunity Area B: Enhancing Envelope Standards and Metadata
Rule Opportunity #3: Expand ongoing payload agnostic approach for explicitly enumerating Payload Types for transactions newly mandated by ACA
Rule Opportunity #4: Explore convergence of Envelope Standards
Rule Opportunity #4A: Explore Suitability of other envelope approaches (e.g., JavaScript Object Notation (JSON))
Opportunity Area C: Enhancing Reliability and Security
Rule Opportunity #5: Reliable and secure handling of attachments
Rule Opportunity #6: Explore convergence of Authentication Standards
Rule Opportunity #7: Explore industry-wide policy for uniform use of digital certificates
Rule Opportunity #8: Explore TLS 1.X as part of base requirement for transport security
Rule Opportunity #9: Explore enhanced envelope level security (e.g., Signature, SAML Authorization), determining B2B nature of transactions and that some signatures may be applied at the document (payload) level.
Opportunity Area D: Exploring Additional Transport Options
Rule Opportunity #10: Explore support for ONC DIRECT as an additional transport option

Rule Opportunity #11: Explore support for Representational State Transfer (REST) as an additional transport option
Rule Opportunity #12: Explore support for Secure File Transfer Protocol (SFTP) as an additional transport option
Opportunity Area E: Specificity Around Message Interaction Requirements
Rule Opportunity #13: Defining Transaction Specific Message Interaction (e.g., Real time, Batch) Requirements

To select the opportunities that would provide the best value to the industry CAQH CORE developed an objective approach using a set of 44 business and technical criteria to evaluate and compare the potential rule opportunities identified for Phase IV CORE Connectivity recognizing that all of the CAQH CORE Rules are expected to evolve in future phases. Some key business and technical criteria among them are that the Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 will:

- not create or promote proprietary approaches to electronic interactions/transactions
- not be based on the least common denominator but rather will encourage feasible progress, promote cost savings, and efficiency
- address both batch and real time processing modes, with a movement towards real time (where/when appropriate)
- be developed using a consensus-based, multi-stakeholder approach
- builds upon existing standards
- be focused on Business to Business (B2B) transactions
- create a base and not a "ceiling"
- be vendor neutral
- be built upon HIPAA, and align with other key industry bodies in order to promote interoperability
- address interest in XML, or other evolving standards where appropriate
- support the Guiding Principles of HHS' Nationwide Health Information Network (now the eHealth Information Exchange[12])

Table 2.2-2 shows the results of applying the business and technical criteria to the potential rule opportunities.

Table 2.2-1 Rule Opportunity Selection

Rule Opportunities to be addressed in Phase IV	Rule Opportunities to be addressed in Phase IV if time allows	Rule Opportunities deferred for future consideration
Rule Opportunity #1: Improve clarity around Real time and Batch requirements, error handling	Rule Opportunity #5: Reliable and secure handling of attachments	Rule Opportunity #4A: Explore Suitability of other envelope approaches (e.g., JavaScript Object Notation (JSON))
Rule Opportunity #2: Address CAQH CORE Connectivity Rule v2.2.0 implementer feedback specific to technical issues	Rule Opportunity #7: Explore industry-wide policy for uniform use of digital certificates	Rule Opportunity #11: Explore support for Representational State Transfer (REST) as an additional transport option
Rule Opportunity #3: Expand ongoing payload agnostic approach for explicitly enumerating Payload Types for transactions newly mandated by ACA	Rule Opportunity #9: Explore enhanced envelope level security (e.g., Signature, SAML Authorization), determining B2B nature of transactions and that some signatures may be applied at the document (payload) level.	Rule Opportunity #12: Explore support for Secure File Transfer Protocol (SFTP) as an additional transport option
Rule Opportunity #4: Explore convergence of Envelope Standards	Rule Opportunity #10: Explore support for ONC DIRECT as an additional transport option	
Rule Opportunity #6: Explore convergence of Authentication Standards		

[12] See §7.2 Abbreviations and Definitions Used in this Rule

Table 2.2-1 Rule Opportunity Selection		
Rule Opportunities to be addressed in Phase IV	**Rule Opportunities to be addressed in Phase IV if time allows**	**Rule Opportunities deferred for future consideration**
Rule Opportunity #8: Explore TLS 1.X as part of base requirement for transport security		
Rule Opportunity #13: Defining Transaction Specific Message Interaction (e.g., Real time, Batch) Requirements		

2.3 Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 Builds on Foundation Established by Previous CAQH CORE Connectivity Rules

2.3.1 Base Minimum Requirements Specified in the Phase I CAQH CORE 153 Connectivity Rule

The Phase I CAQH CORE 153 Connectivity Rule established the requirement to use the HTTP/S secure transport protocol over the public Internet. It also specified a minimum set of metadata that must be outside the ASC X12N payload (e.g., date/time, payload ID, and other elements), and aspects of connectivity/security such as connectivity response times, acknowledgements and errors. The Phase I CAQH CORE 153 Connectivity Rule also established the CORE Connectivity "Safe Harbor" which allows HIPAA covered entities or their agents to implement additional other connectivity/security methods in addition to the requirement to support the CORE Connectivity Rule.

2.3.2 Phase II CAQH CORE 270 Connectivity Rule Specified Robust, Prescriptive Requirements

CAQH CORE was aware the Phase I CAQH CORE 153 Connectivity Rule did not provide the optimum level of specificity for implementations as it was developed as a first step. Voluntary CORE certified implementations were based on many types of message enveloping methods: e.g., 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 existed in field names and locations of Phase I CAQH CORE 153 Connectivity Rule metadata, message envelope structure, submitter authentication methods, routing approaches and security related information. As a result, such variations among enveloping methods and metadata posed a major challenge for interoperability. Therefore, the Phase II CAQH CORE 270 Connectivity Rule specified more prescriptive requirements for message envelopes, message envelope metadata, and submitter authentication methods.

2.3.2.1 Two Message Envelope Standards Specified in CAQH CORE Phase II Rule

Just as paper documents need to be placed in an envelope (container), electronic documents (e.g., eligibility inquiries, electronic claims, etc.) must be placed into a container for electronic transmission from the sender to the receiver. These electronic containers, called message envelopes, must also include the critical information needed to identify the sender, receiver, and other information essential for ensuring the electronic documents in the message envelope are delivered to the intended recipient securely and reliably. For message envelopes the terms for the various pieces of information required are called Message Envelope Metadata specifying the fields and their corresponding values within the message envelope that describe the documents (message payload). A message envelope consists of a well-defined structure for organizing and formatting the message envelope metadata, which also includes other information, such as date, time, unique identifiers for each message envelope to enable reliable tracking and auditing.

The Phase II CAQH CORE 270 Connectivity Rule further facilitated interoperability by requiring the use of two message envelope standards that were shown to meet the agreed upon Phase II CAQH CORE Connectivity

criteria, have significant installed base in the healthcare industry, and perform well under real world transaction loads. These two envelope standards were HTTP MIME Multipart and SOAP + WSDL.

Since both these standards have significant merits, the advantages and challenges of having a single envelope standard versus both of these envelope standards as part of the Phase II CAQH CORE 270 Connectivity Rule was debated. 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 compliant with the Phase II CAQH CORE 270 Connectivity Rule. Since both standards met the criteria and had large installed bases, CAQH CORE determined that convergence on a single standard in Phase II would create a barrier to adoption of the Phase II CAQH CORE 270 Connectivity Rule by a large segment of the industry.

2.3.2.2 CAQH CORE Phase II Rule Specified Two Submitter Authentication Methods

HIPAA Security regulations[13] at 45 CFR §164.304 Definitions define "authentication as the corroboration that a person [entity] is the one claimed" and further identifies that the "*Technical safeguards* are the technology and the policy and procedures for its use that protect electronic protected health information and control access to it." O'Reilly[14] goes on to further describe authentication as "The process of proving that a subject (e.g., a user or a system) is what the subject claims to be. Authentication is a measure used to verify the eligibility of a subject and the ability of that subject to access certain information. It protects against the fraudulent use of a system or the fraudulent transmission of information. There are three classic ways to authenticate oneself: something you know, something you have, and something you are."

Thus, it is essential to validate a particular entity's identity for granting access to sensitive data or functionalities contained within the system. One of the most common authentication methods used in general today is a Username+Password. Digital certificates are another commonly used method and are considered to "provide the most secure means of authenticating identities."[15] Each authentication method has advantages and disadvantages in terms of security, usability, and breadth of support. Password-based authentication methods, however, do not provide strong security.

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, the Phase II CAQH CORE 270 Connectivity Rule required only server-side implementations to support one of two submitter authentication methods:

- Username + Password
- X.509 Digital Certificate

2.4 Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 - Key Enhancements relative to Phase II CAQH CORE Connectivity Rule

To address the problems described in §2.1 and to advance the vision for future phases that was identified in Phase II and Phase III, this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 has key enhancements by converging on single envelope and authentication standard, improving transaction support for the 3rd set of ACA mandated transactions, and by improving robustness and security. These enhancements are described below.

---

[13] 68 FR 8376, Feb. 20, 2003, as amended at 74 FR 42767, Aug. 24, 2009; 78 FR 5693, Jan. 25, 2013
[14] D. Russell and G.T. Gangemi Sr., "Computer Security Basics", O'Reilly & Associates, Inc., 1992
[15] Centers for Medicare & Medicaid Services, Enterprise Information Security Group, Risk Management Handbook volume III Standard 3.1, CMS Authentication Standards, Final Version 1.3, April 17, 20014

2.4.1 Convergence on a Single Message Envelope Standard

The Phase II CAQH CORE 270 Connectivity Rule identified convergence to single envelope standard as a vision for future phases of connectivity based on greater industry experience with implementing the two message envelope standards specified in the rule.

After extensive analysis CAQH CORE determined that converging on the use of SOAP+WSDL as the single message envelope standard in this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 includes these benefits:

- limits variations in use of SOAP for real time and batch processing modes by requiring the use of MTOM for both processing modes
- relatively simple rule change
- significant ROI through improvements in interoperability
- simplicity of rule requirements
- reduction of implementation cost and complexity by having fewer options
- XML based and therefore extensible
- good tooling support for SOAP in most platforms
- alignment with clinical initiatives and industry trends

2.4.2 Convergence on a Single Submitter Authentication Method

The Phase II CAQH CORE 270 Connectivity Rule identified convergence to single authentication standard as a vision for future phases of connectivity based on greater industry experience with implementing the two message authentication standards specified in the rule.

After extensive analysis CAQH CORE determined that converging on the use of the X.509 digital certificate as the single authentication standard in this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 includes these benefits:

- relatively simple rule change
- significant ROI through improvements in interoperability
- simplicity of rule requirements
- reduction of implementation cost and complexity by having fewer options
- X.509 Client Certificate based authentication over SSL/TLS is significantly more secure than Username+Password
- alignment with clinical initiatives and industry trends

2.4.3 Enhancements to Message Interactions

The Phase II CAQH CORE 270 Connectivity Rule defined message interactions for conducting Real Time and Batch interactions. Phase IV preserves the Real Time and Batch interactions while adding some message interactions that could be used as generic building blocks for supporting current or future transactions. The message interactions for the 3rd set of ACA mandated transactions are illustrated in Section 7 using Uniform Markup Language (UML) sequence diagrams, also known simply as sequence diagrams.

A sequence diagram is an interaction diagram used to visualize how a client (submitter) and a server (receiver) operate with one another and in what order for the transactions addressed by this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0. Some interactions are scenarios in which the business transaction (message payload) is to be processed in real time by the server while other interactions are scenarios in which the business transaction(s) (message payload) are to be processed as a batch after the server has successfully received the batch and the communication session has ended. When an interaction includes multiple client requests and server responses, e.g., a batch of health care claims, each pair of interactions and its corresponding (synchronous)

response is shown in the sequence diagram. The UML sequence diagrams in this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 are specific to the HIPAA mandated transactions to which this rule applies.

2.4.4 Improved Support for Security Compliance and Stronger Security

The Phase II CAQH CORE 270 Connectivity Rule v2.2.0 requires the implementation of the Secure Sockets Layer (SSL) v3.0 as a minimum while optionally allowing entities to implement the Transport Layer Security (TLS) v1.0 or higher when an entity is required to comply with the National Institute of Standards and Technology (NIST) Federal Information Processing Standards (FIPS 140).

Analysis conducted by CAQH CORE indicated that while SSL v3.0 is commonly used in the industry, some HIPAA covered entities or their agents (e.g., Federal government trading partners, eHealth Exchange) are required to also comply with the FIPS 140-2, which essentially prohibits the use of SSL v3.0 and TLS v1.0. As per NIST 800-52r1, federal government entities are required to implement TLS 1.1 or higher. Relative to SSL v3.0, TLS 1.1 and to a larger extent, TLS 1.2 has improvements in security for data in transit (e.g., in message integrity, encryption algorithms, and key generation). However, platform and programming support for, and industry experience in implementing TLS 1.1 and TLS 1.2 is limited at this time. Considering this, this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 (See §3.2) strikes a balance between the need to accommodate HIPAA covered entities or their agents that must also comply with FIPS 140-2, while allowing non-government entities to continue using non-FIPS compliant security at the transport security layer as well as at the message envelope security layer.

Further, by allowing the use of TLS 1.1 or higher in lieu of SSL v3.0 for both FIPS 140-2 compliance and for the sake of stronger security, this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 (See §3.2) enables a transition path from SSL v3.0 to TLS 1.1 or higher.

2.4.5 CAQH CORE Process for Maintaining Processing Mode & Payload Type Specifications

Processing modes or computing modes are classifications of different types of computer processing, e.g., batch, real time.[16] In the context of CAQH CORE Operating rules, the concept of processing mode applies to the timeframe within which a receiver of a payload of transactions processes those transactions and returns to the sender a payload of appropriate acknowledgements.

A message payload is the essential data that is being routed between a sender and a receiver during a connectivity session. In the context of this CAQH CORE Operating rule, a payload could be one or more healthcare claims or referral requests, etc. In order to enable efficient and effective handling of the various kinds of payloads that could be exchanged, a unique "payload type identifier" is assigned to each kind of payload.

As this rule becomes widely adopted and implemented in health care the experience and learning gained from implementers CAQH CORE recognizes there may be a need to modify either the Phase IV CAQH CORE Processing Mode requirements or the Phase IV CAQH CORE Payload Types or both in order to be agile and flexible in meeting emerging or new industry needs. To meet this anticipated need to enable review and maintenance of the processing modes for the administrative transactions addressed by this rule and payload type identifiers are specified in a separate companion document to this rule. A process and policy to address the review and maintenance will be developed by CAQH CORE. (See §3.7.3)

The Phase IV CAQH CORE-required Payload Types Table includes payload type values for all HIPAA mandated ASC X12N v5010 transactions, including those transactions that are addressed in the Phase I and II CAQH CORE Operating Rules for eligibility, claim status and the Phase III CAQH CORE Operating Rules for ERA. While HIPAA covered entities or their agents are required to use this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 for the exchange of claims, prior authorization, benefit enrollment and maintenance, and health plan

[16] See §7.2 Abbreviations and Definitions Used in this Rule

premium payment transactions, subject to the Safe Harbor provisions of the Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 HIPAA covered entities or their agents may also use this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 for the exchange of eligibility, claim status and ERA transactions in accordance with the Safe Harbor provision of the Phase II CAQH CORE 270 Connectivity Rule version 2.2.0. However, this does not permit any HIPAA covered entity or its agent to discontinue support for the exchange of the eligibility, claim status and ERA transactions as required in the Phase II CAQH CORE 270 Connectivity Rule version 2.2.0. HIPAA covered entities or their agents may also use this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 for the exchange of ASC X12N transactions not mandated by HIPAA.

2.4.6 Backward Compatibility with Phase I and II CAQH CORE Connectivity Rules

CAQH CORE thoroughly examined maintaining *backward compatibility* with the Phase I and Phase II CAQH CORE Connectivity Rules while also evolving the Phase IV CAQH CORE 470 Connectivity Operating Rule v4.0.0, which is applicable to a different set of administrative transactions[17]. These ACA-mandated CAQH CORE operating rules currently remain in effect and cannot be modified by Phase IV CAQH CORE Operating Rules. In general, the concept of *backward compatibility* in relationship to technical specifications means that implementers of a newer version of a specification will be able to interact and interoperate with implementers of a previous version easily and without major modifications to either version.

In the context of this Phase IV CAQH CORE 470 Connectivity Operating Rule v4.0.0 *backward compatibility* means that key requirements to support two message envelope standards and two submitter authentication methods specified in previous versions of CAQH CORE Connectivity Operating Rules would become an impediment to realizing some of the Phase IV high priority rule opportunities agreed to by CAQH CORE. (See §2.2 Table 2.2-2.) Since this Phase IV CORE Connectivity Rule is intended to be independent of the current ACA-mandated CAQH CORE Operating Rules and must stand alone on its own merits implementers of those rules are not required to de-implement or otherwise discontinue support for any of those rules.

Further, as HIPAA covered entities or their agents may also use this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 for the exchange of transactions addressed by previous phases (Phase I and II) in accordance with the Safe Harbor provision of the Phase II CAQH CORE 270 Connectivity Rule version 2.2.0, the improvements made in this Phase IV CORE Connectivity Rule can also benefit those transactions. However, this does not permit any HIPAA covered entity or its agent to discontinue support for the exchange of transactions addressed in CORE Phase II and CORE Phase III as required in the Phase II CAQH CORE 270 Connectivity Rule version 2.2.0.

3 SCOPE

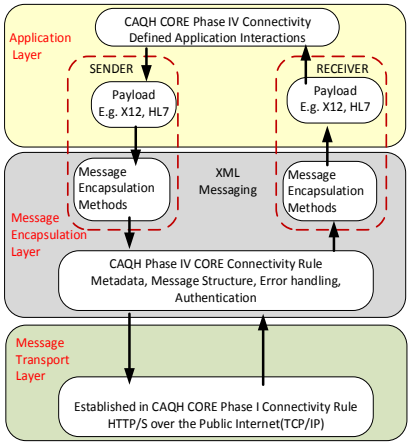
3.1 What the Rule Applies To

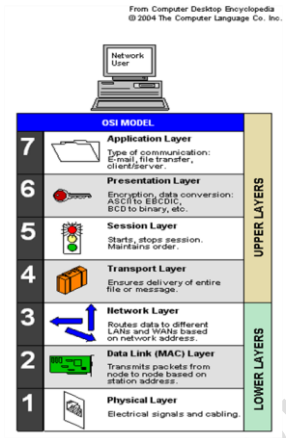
The technical scope of this Phase IV CORE Connectivity Rule can be described in terms of the specific network layers within the Open Systems Interconnection Basic Reference Model[18] (OSI model). As shown in the diagram below, the scope of Phase IV CORE Connectivity Rule is OSI Layers 3 and 4 (Transport and Network layers) and OSI Layers 5 and 6 (Session and Presentation layers, also called Message Encapsulation layers).

[17] The first set of operating rules under ACA Section 1104 applies to eligibility and claim status transactions; these operating rules were effective 01/01/13. The second set of operating rules applies to EFT and ERA; these operating rules were effective 01/01/14.

[18] 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 #3.1.1





OSI Model		Messaging Infrastructure Model
Application Layer	OSI 7	Application Layer
Presentation Layer	OSI 6	Message Encapsulation Layer (envelope)
Session Layer	OSI 5	
Transport Layer	OSI 4	Message Transport Layer
Network Layer	OSI 3	

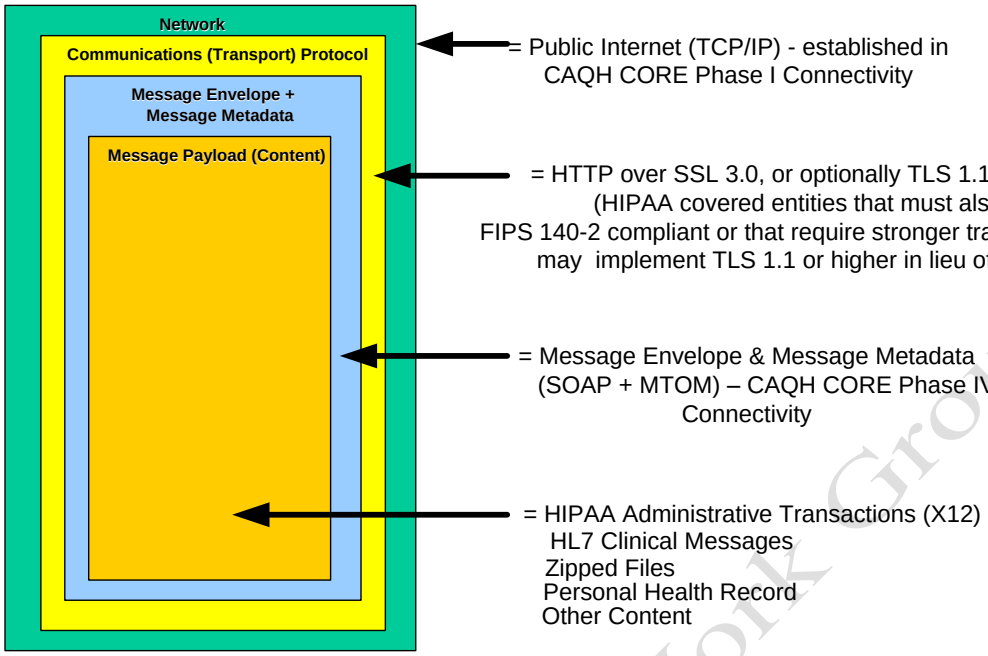
Figure Notes:

- CAQH CORE Phase IV Connectivity Rule addresses Layers 5 and 6 of the OSI Model
- Layer 3 and 4, the Transport Layer and Network Layer, was established as HTTP/S over the public internet in the CAQH CORE 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.

As shown in the Figure 3.1.1 above, typically an application file (or Payload) such as ASC 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 handles 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 #3.1.2 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 CAQH CORE 153 Connectivity Rule version 1.1.0 established the CAQH CORE foundational 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 X12N), HL7 clinical messages, zipped files, etc.

Figure #3.1.2



3.2 Standards Used in this Rule

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

- HTTP Version 1.1[19]
- SSL Version 3.0.
    - This does not preclude the optional use of TLS 1.1 (or a higher version) for connectivity with trading partners that require FIPS 140-2 compliance or whose security policies require the enhanced security afforded by TLS 1.1 or higher. Entities that must also be FIPS 140-2 compliant or whose security policies require enhanced security may implement TLS 1.1 or higher in lieu of SSL 3.0.
- SOAP Version 1.2
- WSDL Version 1.1

3.3 When the Rule Applies

The Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 applies when trading partners are exchanging any transaction specified in the third set of the Affordable Care Act (ACA) §1104 administrative transactions, i.e.:

- ASC X12N v5010 837 Claim
- ASC X12N v5010 278 Request and Response

[19] Hereafter the combination of HTTP and SSL/TLS is referenced as HTTP/S.

- ASC X12N v5010 820
- ASC X12N v5010 834

The Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 may also be applied to other payload types. Note: some entities may also apply this rule to other ASC X12N administrative transactions. This Phase IV CORE Connectivity Rule is a Safe Harbor (See §5), 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. Further, HIPAA covered entities or their agents may also use this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 for the exchange of eligibility, claim status and ERA transactions in accordance with the Safe Harbor provision of the Phase II CAQH CORE 270 Connectivity Rule version 2.2.0. However, this does not permit any HIPAA covered entity or its agent to discontinue support for the exchange of transactions addressed in CORE Phase II and CORE Phase III as required in the Phase II CAQH CORE 270 Connectivity Rule version 2.2.0.

3.4 When the Rule Does Not Apply

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

- When HIPAA covered entities or their agents exchange payloads other than
  - ASC X12N v5010 837 Claim
  - ASC X12N v5010 278 Request and Response
  - ASC X12N v5010 820
  - ASC X12N v5010 834

This rule does not address requirements for the use of the ASC X12N v5010 820 and the ASC X12N v5010 834 transactions by the ACA Federal or state Health Information Exchanges (HIX).

This rule is designed to be payload agnostic, and as such it is expected that HIPAA covered entities or their agents will use this methodology for other payloads as described in §3.3; however, the rule does not require this.

3.5 What the Rule Does Not Require

The Phase IV CORE Connectivity Rule (See §5):

- **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-compliant method for all new connections.
- **DOES NOT** require that all CORE trading partners use only one method for all connections.
- **DOES NOT** require any HIPAA covered entity or its agent to do business with any trading partner or other HIPAA covered entity or its agent.

Further, the Phase IV 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.

3.6 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).
- 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 §4.2.6.2)

3.7 CORE-required Processing Mode and Payload Type Tables

This Phase IV CORE Connectivity Rule is comprised of the complete rule itself, which specifies all rule requirements; and a companion document to the rule, which specifies additional rule requirements addressing Phase IV CAQH CORE-required processing modes and payload type tables. This enables the necessary flexibility to review and maintain the processing modes and payload types based on Federal regulation or Federal notices to the industry impacting the transactions addressed by this rule.

3.7.1 CORE-required Processing Mode Table

The Phase IV CAQH CORE-required Processing Mode Table (see §4.4.3) specifies the comprehensive and normative processing mode requirements (i.e., Real Time and/or Batch) for the transactions addressed by this rule.

3.7.2 CORE-required Payload Type Table

The Phase IV CAQH CORE-required Payload Type Table (see §4.4.3) specifies the comprehensive and normative identifiers for the CORE Envelope Metadata Payload Type Element as defined in the Table of CORE Envelope Metadata. (See §4.4.2.)

The Payload Type identifiers specified in the Phase IV CAQH CORE-required Payload Type Table apply when an entity is exchanging the transactions addressed by this rule in conformance with the requirements specified in §4 and subsections.

3.7.3 Maintenance of the CORE-required Processing Mode and Payload Type Tables

CAQH CORE recognizes that as this rule becomes widely adopted and implemented in health care the experience and learning gained from implementers may indicate a need to modify either the Phase IV CAQH CORE-required Processing Mode Table or the Phase IV CAQH CORE-required Payload Type Table or both to meet emerging or new industry needs. Given this anticipated need a process and policy to enable the review and maintenance of these tables specified in the companion document to this rule, *COREProcessingModePayloadTypeTables.docx,* will be developed by CAQH CORE.

Such review and maintenance of either the Phase IV CAQH CORE-required Processing Mode Table or the Phase IV CAQH CORE-required Payload Type Table or both will follow standard CAQH CORE processes for rule revisions. CAQH CORE will develop such a process and policy for the first review of potential revisions of these tables in accordance with CAQH CORE Guiding Principles following the approval of the Phase IV CAQH CORE Operating Rules. The first review may commence

- One year after the passage of a Federal regulation requiring implementation of this CAQH CORE rule

Or

- When Federal regulation or Federal notices to the industry impacting the transactions addressed by this rule are published.

Substantive changes necessary to the tables will be reviewed and approved by CAQH CORE as necessary to ensure accurate and timely revision. The impact of any such changes to any Phase IV CAQH CORE Infrastructure rule will be considered during the review of potential revisions. Phase IV CAQH CORE Infrastructure rules address other requirements for conducting the transactions addressed by this rule, such as response times for Real Time and/or Batch, System Availability, Companion Document flow and format, etc.

3.8 How This Rule Relates to Previous CAQH CORE Operating Rules

The Phase I CAQH CORE 153 Connectivity Rule established the required use of the public Internet. The Phase II CAQH CORE 270 Connectivity Rule extended the Phase I CAQH CORE 153 Connectivity Rule by establishing a Safe Harbor and specifying the connectivity that all HIPAA or their agents covered entities must implement and support. (See §5) Each of the Phase I and Phase II CAQH CORE rule requirements has been incorporated into this Phase IV Rule except that the MIME Multipart envelope and Username+Password submitter authentication requirements are not retained in this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0. The use of MTOM for SOAP Real Time in this phase implies that use of CDATA tags for SOAP Real Time inline payload, or use of Base64 encoding for payloads with non-printable characters are not a requirement in this phase. Further, relative to Phase II, the SSL/TLS requirements in this phase have been updated (§3.2) to improve support for FIPS 140-2 compliance.

Since this Phase IV CORE Connectivity Rule is intended to be independent of both the Phase I CAQH CORE 153 Connectivity Rule and the Phase II CAQH CORE 270 Connectivity Rule and must stand alone on its own merits; implementers of those rules are not required to de-implement or otherwise discontinue support for any of these Phase I and/or Phase II CAQH CORE rules requirements.

While this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 is mandated for the exchange of transactions addressed in Phase IV CAQH CORE, it may also be used for the exchange of transactions addressed by the previous CORE Phases (i.e., Phases I, II and III) in accordance with the Safe Harbor provision of the Phase II CAQH CORE 270 Connectivity Rule version 2.2.0. However, this does not permit any HIPAA covered entity or its agent to discontinue support for the exchange of transactions addressed in CORE Phase II and CORE Phase III as required in the Phase II CAQH CORE 270 Connectivity Rule version 2.2.0.

3.9 Assumptions

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.
- 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 IV CAQH CORE Rules; as such, all the CORE Guiding Principles apply to this rule and all other rules.
- All entities seeking voluntary Phase IV certification will be Phase I, Phase II and Phase III certified, or concurrently testing for compliance with these rules as they provide a foundation for Phase IV CAQH CORE. The exception is vendors/clearinghouses that do not conduct the ASC X12N v5010 270/271 eligibility, the ASC X12N v5010 276/277 claim status or the ASC X12N v5010 835 remittance advice transactions.

4 RULE

This section specifies the requirements for transport, message envelope, submitter authentication, envelope metadata and the specifications for SOAP+WSDL. The rationale and business justification for these conformance requirements are described in §2.

4.1 CORE Message Envelope and Submitter Authentication Requirements

This rule requires HIPAA covered entities or their agents to support only one set of requirements for message enveloping and one method for submitter authentication in order to reduce variations and enable greater interoperability in the market.)

4.1.1 Message Envelope Requirement

This rule requires the use of SOAP+WSDL (See §4.1.3).

4.1.2 Submitter Authentication Requirement

This rule requires the use of X.509 Client Authentication (mutual authentication) over SSL 3.0 (TLS 1.1 or higher may be used as per the specifications in §3.2).

4.1.3 Specifications for SOAP+WSDL Envelope Standard (normative[20])

This section defines the SOAP+WSDL envelope method for Phase IV CAQH CORE 470 Connectivity Rule v4.0.0. 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 §7.2: Abbreviations and Definitions used in this Rule*.

4.1.3.1 Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 XML Schema Specification (normative)

The Phase IV CAQH CORE compliant XML Schema Specification file name below is called *CORERule4.0.0.xsd*, and is available at *http://www.caqh.org/SOAP/WSDL/CORERule4.0.0.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*)

[20] See §7.2 Abbreviations and Definitions used in this Rule for a definition of Normative.

- Batch Results Acknowledgement Submission (Element name is *COREEnvelopeBatchResultsAckSubmission*)
- Batch Results Acknowledgement Submission Response (Element name is *COREEnvelopeBatchResultsAckSubmissionResponse*)

A consequence of the CAQH CORE XML Schema Specification being normative is that any changes to the structure and syntax of the SOAP Body make the implementation non-compliant. Any such implementations must be done under the CORE Safe Harbor provision.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.caqh.org/SOAP/WSDL/CORERule4.0.0.xsd"
targetNamespace="http://www.caqh.org/SOAP/WSDL/CORERule4.0.0.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:base64Binary" 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: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="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>
```

4.1.3.2 CORE Connectivity Web Services Definition Language (WSDL) Specification (normative)

The Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 Web Services Definition Language (WSDL) file below is called *CORERule4.0.0.wsdl*, and is available at *www.caqh.org/SOAP/WSDL/CORERule4.0.0.wsdl*. The WSDL below makes use of the XML Schema (*CORERule4.0.0.xsd*) as specified in §4.1.3.1. 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 Submission Transaction (Operation name is *GenericBatchSubmissionTransaction*)
- Generic Batch Submission Acknowledgment Retrieval Transaction (Operation name is *GenericBatchSubmissionAckRetrievalTransaction*)
- Generic Batch Retrieval Transaction (Operation name is *GenericBatchRetrievalTransaction)*
- Generic Batch Receipt Confirmation Transaction (Operation name is *GenericBatchReceiptConfirmationTransaction)*

The CAQH CORE Connectivity WSDL uses an implicit style of specification, which allows the optional use of additional elements within the SOAP Header. Server entities that require the use of SOAP Header elements must define their use in the entity's Connectivity Companion Document. Client or Server entities that do not use these SOAP Header elements must ignore them.

```
<?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/CORERule4.0.0.xsd"
                              xmlns="http://schemas.xmlsoap.org/wsdl/"
                              name="CORE"
                              targetNamespace="http://www.caqh.org/SOAP/WSDL/">

  <!-- TYPES (BEGIN) -->
  <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/CORERule4.0.0.xsd"
schemaLocation="CORERule4.0.0.xsd"/>
    </xsd:schema>
  </wsdl:types>
  <!-- TYPES (END) -->

  <!-- MESSAGE (BEGIN) -->
  <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>
 <!-- MESSAGE (END) -->

 <!-- PORTTYPE (BEGIN) -->
 <wsdl:portType name="CORETransactions">

   <!-- OPERATION: REAL TIME INTERACTION (BEGIN) -->
   <wsdl:operation name="RealTimeTransaction">
     <wsdl:input message="CORE:RealTimeRequestMessage"/>
     <wsdl:output message="CORE:RealTimeResponseMessage"/>
   </wsdl:operation>
   <!-- OPERATION: REAL TIME INTERACTION (END) -->

   <!-- OPERATION: BATCH INTERACTION (BEGIN) -->
   <wsdl:operation name="BatchSubmitTransaction">
     <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>
   <!-- OPERATION: BATCH INTERACTION (END) -->

   <!-- OPERATION: GENERIC PUSH (BEGIN) -->
   <wsdl:operation name="GenericBatchSubmissionTransaction">
     <wsdl:input message="CORE:BatchSubmissionMessage"/>
     <wsdl:output message="CORE:BatchSubmissionResponseMessage"/>
   </wsdl:operation>
    <wsdl:operation name="GenericBatchSubmissionAckRetrievalTransaction”>
      <wsdl:input message="CORE:BatchSubmissionAckRetrievalRequestMessage"/>
```

```
      <wsdl:output message="CORE:BatchSubmissionAckRetrievalResponseMessage”/>
    </wsdl:operation>
    <!-- OPERATION: GENERIC PUSH (END) -->

    <!-- OPERATION: GENERIC PULL (BEGIN) -->
    <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>
    <!-- OPERATION: GENERIC PULL (END) -->
  </wsdl:portType>
  <!-- PORTTYPE (END) -->

  <!-- BINDING (BEGIN) -->
  <wsdl:binding name="CoreSoapBinding" type="CORE:CORETransactions">
    <soap12:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>

    <!-- OPERATION: REAL TIME TRANSACTION (BEGIN) -->
    <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>
    <!-- OPERATION: REAL TIME TRANSACTION (END) -->

    <!-- OPERATION: BATCH TRANSACTION (BEGIN) -->
    <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="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>
    <!-- OPERATION: BATCH TRANSACTION (END) -->

    <!-- OPERATION: GENERIC PUSH (BEGIN) -->
    <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="GenericBatchSubmissionAckRetrievalTransaction">
     <soap12:operation soapAction="GenericBatchSubmissionAckRetrievalTransaction" style="document"/>
      <wsdl:input>
        <soap12:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap12:body use="literal"/>
      </wsdl:output>
    </wsdl:operation>
    <!-- OPERATION: GENERIC PUSH (END) -->

    <!-- OPERATION: GENERIC PULL (BEGIN) -->
    <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>
    <!-- OPERATION: GENERIC PULL (END) -->

  </wsdl:binding>
  <!-- BINDING (END) -->

  <!-- SERVICE (BEGIN) -->
  <wsdl:service name="Core">
    <wsdl:port name="CoreSoapPort" binding="CORE:CoreSoapBinding">
      <soap12:address location="http://URL OF WEB SERVICE"/>
    </wsdl:port>
  </wsdl:service>
  <!-- SERVICE (END) -->

</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[21]. They are based on the real-world examples provided by CAQH CORE participants, but have been updated to use the CAQH CORE-required metadata that is part of Phase IV CAQH CORE 470 Connectivity Rule v4.0.0.

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

4.1.3.3 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.

This shows the following components:

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

```
POST /CORE/PriorAuthRealTime HTTP/1.1
Host: server_host:server_port
Content-Type: multipart/related; boundary= MIMEBoundaryurn uuid 5117AAE1116EA8B87A1200060184614;
type="application/xop+xml"; start="0.urn:uuid:5117AAE1116EA8B87A1200060184615"; start-
info="application/soap+xml";
action="http://www.caqh.org/SOAP/WSDL/CORETransactions/RealTimeTransaction"

--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>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Body>
              <ns1:COREEnvelopeRealTimeRequest
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule4.0.0.xsd">
                <PayloadType>X12_278_Request_005010X217E1_2</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>4.0.0</CORERuleVersion>
                    <Payload>
                      <xop:Include href="cid:1.urn:uuid:5117AAE1116EA8B87A1200060184692"
                    xmlns:xop="http://www.w3.org/2004/08/xop/include" />
                    </Payload>
              </ns1:COREEnvelopeRealTimeRequest>
                </soapenv:Body>
</soapenv:Envelope>

--MIMEBoundaryurn uuid 5117AAE1116EA8B87A1200060184614
Content-Type: image/jpeg
Content-Transfer-Encoding: binary
Content-ID: <1.urn:uuid:5117AAE1116EA8B87A1200060184692>

<Real Time Request Payload (e.g., a payload of type X12_278_Request_005010X217E1_2) goes here>

--MIMEBoundaryurn uuid 5117AAE1116EA8B87A1200060184614--
```

4.1.3.4 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 Phase IV CORE Connectivity Rule. (See §4.4)

```
HTTP/1.1 200 OK
Content-Type: multipart/related; boundary= MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614;
type="application/xop+xml"; start="0.urn:uuid:5117AAE1116EA8B87A1200060184615"; start-
info="application/soap+xml";
action="http://www.caqh.org/SOAP/WSDL/CORETransactions/RealTimeTransactionResponse"

--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>

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
       <soapenv:Body>
              <ns1:COREEnvelopeRealTimeResponse
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule4.0.0.xsd">
                     <PayloadType>X12 278 Response 005010X217E1 2</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>4.0.0</CORERuleVersion>
                      <Payload>
                    <xop:Include href="cid:1.urn:uuid:5117AAE1116EA8B87A1200060184692"
                    xmlns:xop="http://www.w3.org/2004/08/xop/include" />
                      </Payload>
                     <ErrorCode>Success</ErrorCode>
                     <ErrorMessage></ErrorMessage>
              </ns1:COREEnvelopeRealTimeResponse>
       </soapenv:Body>
  </soapenv:Envelope>

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

<Real Time Response Payload (e.g., a payload of type X12_278_Response_005010X217E1_2) goes here>

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```

4.1.3.5 Batch Submission Message (non-normative)[22]

The Batch Submission message structure shown below specifies SOAP 1.2, and also uses MTOM (See §7.1) to send the payload file. 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 Phase IV CORE Connectivity Rule. (See §4.4)
3. The Batch file (MTOM attachment) is shown colored in orange.

[22] The Batch Payload Submission in a Generic Push interaction (i.e., Step 1 in the sequence diagram shown in §7.3.3.1) uses the same request message as the Batch Submission Request message structure depicted below, with *PayloadType* values based on what is being submitted.

```
POST /CORE/PriorAuthBatch 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:Body>
              <ns1:COREEnvelopeBatchSubmission
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule4.0.0.xsd">
                     <PayloadType>X12_278_Request_005010X217E1_2</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>4.0.0</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>

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```

4.1.3.6 Batch Submission Response Message (non-normative)[23]

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 IV Connectivity Rule. (See §4.4)

[23] The response to Batch Payload submission in a Generic Push interaction (i.e., Step 2 in the sequence diagram in §7.3.3.1) 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.

586

```
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/CORERule4.0.0.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>4.0.0</CORERuleVersion>
                      <ErrorCode>Success</ErrorCode>
                      <ErrorMessage></ErrorMessage>
                </ns1:COREEnvelopeBatchSubmissionResponse>
         </soapenv:Body>
  </soapenv:Envelope>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339--
```

587

588 *4.1.3.7 Batch Submission Acknowledgement Retrieval Request Message (non-normative)*

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

592 1. The HTTP Headers are shown colored in blue.

593 2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE
594 Phase IV Connectivity Rule. (See §4.4)

595

596

```
POST /CORE/PriorAuthBatch 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:Body>
              <ns1:COREEnvelopeBatchSubmissionAckRetrievalRequest
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule4.0.0.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>4.0.0</CORERuleVersion>
              </ns1:COREEnvelopeBatchSubmissionAckRetrievalRequest>
         </soapenv:Body>
  </soapenv:Envelope>
--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```

597

598 *4.1.3.8 Batch Submission Acknowledgement Retrieval Response Message (non-normative)*[24]

599 The Batch Submission Acknowledgement Retrieval Response message structure shown below specifies SOAP
600 1.2 and MTOM. This shows the following:

601 1. The HTTP Headers are shown colored in blue.

602 2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE
603 Phase IV Connectivity Rule. (See §4.4)

604 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/CORERule4.0.0.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>
```

[24] Although this example shows an ASC X12 v5010 999 payload type being sent as a response from a server to the client, this could also include an ASC X12 v5010 TA1. Alternatively, the server may elect to send only an ASC X12 v5010 TA1 without any functional group.

```
                    <CORERuleVersion>4.0.0</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--
```

4.1.3.9 Batch Results Retrieval Request Message (non-normative) [25]

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:

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 IV Connectivity Rule. (See §4.4)

```
POST /CORE/PriorAuthBatch 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:Body>
               <ns1:COREEnvelopeBatchResultsRetrievalRequest
               xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule4.0.0.xsd">
                 <PayloadType>X12 278 Request Batch Results 005010X217E1 2</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>4.0.0</CORERuleVersion>
                </ns1:COREEnvelopeBatchResultsRetrievalRequest>
        </soapenv:Body>
  </soapenv:Envelope>
--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```

[25] The Batch Payload retrieval within a Generic Pull interaction (i.e., step 1 in the sequence diagram in §7.3.3.2) 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.

614 *4.1.3.10Batch Results Retrieval Response Message (non-normative)* [26]

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

617 1. The HTTP Headers are shown colored in blue.

618 2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE
619 Phase IV Connectivity Rule. (See §4.4)

620 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/CORERule4.0.0.xsd">
                    <PayloadType>X12_278_Response_005010X217E1_2</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>4.0.0</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--
```

621

622

[26] The Batch Payload retrieval within a Generic Pull interaction (i.e., step 1 in the sequence diagram in §7.3.3.2) 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.

623

624 *4.1.3.11Batch Results Acknowledgement Submission Message (non-normative)*[27]

625 The Batch Results Acknowledgement Submission message structure shown below specifies SOAP 1.2, and also
626 uses MTOM (See §7.2) to send the payload file. This shows the following:

627 1. The HTTP Headers are shown colored in blue.

628 2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CORE
629 Phase IV Connectivity Rule. (See §4.4)

630 3. The Batch file (MTOM attachment) is shown colored in orange.

```
POST /CORE/PriorAuthBatch 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:Body>
              <ns1:COREEnvelopeBatchResultsAckSubmission
              xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERule4.0.0.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>4.0.0</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--
```

631

632

[27] The acknowledgment submission within a Generic Pull interaction (i.e., step 3 in the sequence diagram in §7.3.3.2) uses the same request message as the Batch Results Acknowledgement Submission message structure depicted below, with different *PayloadType* values as appropriate.

633

634 *4.1.3.12Batch Results Acknowledgement Submission Response Message (non-normative)*[28]

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

637 1. The HTTP Headers are shown colored in blue.

638 2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the Phase IV
639 CORE 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/CORERule4.0.0.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>4.0.0</CORERuleVersion>
                      <ErrorCode>Success</ErrorCode>
                      <ErrorMessage></ErrorMessage>
                 </ns1:COREEnvelopeBatchResultsAckSubmissionResponse>
         </soapenv:Body>
  </soapenv:Envelope>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339--
```

640

641

[28] The response to the acknowledgment submission within a Generic Pull interaction (i.e., step 4 in the sequence diagram in §7.3.3.2) uses the same response message as the Batch Results Acknowledgement Submission Response message structure depicted below, with different *PayloadType* values as appropriate.

642

643 *4.1.3.13Error Message Structure (non-normative)*

644 The Error message structure shown below uses the SOAP Fault specifications within SOAP 1.2. As described in
645 §4.2.4, SOAP Faults must be used to send errors at the SOAP level. The HTTP Headers are shown colored in
646 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>
```

647

648 *4.1.3.14Envelope Processing Error Message (non-normative)*

649 The Error message structure shown below illustrates a SOAP-based message that indicates an error has occurred
650 within processing the envelope. The HTTP Headers are shown colored in blue. The remainder of the request is the
651 SOAP Envelope. The envelope structure and metadata that is defined within Phase IV CORE Connectivity Rule is
652 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/CORERule4.0.0.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>4.0.0</CORERuleVersion>
                     <Payload></Payload>
                    <ErrorCode>VersionMismatch</ErrorCode>
                     <ErrorMessage>Expecting Version X, received Version Y</ErrorMessage>
                </ns1:COREEnvelopeRealTimeResponse>
         </soapenv:Body>
  </soapenv:Envelope>
```

653

4.1.4 Real Time and Batch Payload Attachment Handling

Payload must be sent as an MTOM[29] encapsulated object.

4.2 General Specifications Applicable to the SOAP Envelope Method

4.2.1 Required Transport Method

HIPAA covered entities or their agents must be able to implement HTTP/S Version 1.1 over the public Internet as a transport method for the third set of the ACA Section 1104 required transactions as specified in §1 of this rule. Receivers (servers) must be able to perform the role of an HTTP/S server, while senders (clients) must be able to perform the role of an HTTP/S client. By using the HTTP/S protocol, all information exchanged between the sender (client) and receiver (server) is encrypted by a session-level private key negotiated at connection time.

4.2.2 Request and Response Handling

HTTP/S supports a request-response message pattern, meaning that the sender (client) submits a message and then waits for a response from the message receiver (server). This works well for the submission of ASC X12N messages in both Batch and Real Time processing modes, but the response message from the receiver (server) is different depending on whether the sender's (client's) message is a Real Time request, Batch submission, or Batch request pickup.

4.2.3 Real Time Requests

Real Time requests must include a single inquiry or submission as specified in the transaction's corresponding Phase IV CAQH CORE Infrastructure Rule. In this processing mode the response from the message receiver (server) is either

- A transport or message envelope error response (See §4.2.6)

Or

- The corresponding ASC X12 message response (e.g. ASC X12C 005010X231A1 Implementation Acknowledgement for Health Care Insurance (999) [hereafter ASC X12C v5010 999]

Or

- The corresponding ASC X12N v5010 response transaction to the submitted request

4.2.4 Batch Submission

Batch requests are sent in the same way as Real Time requests. In this processing mode the response will differ because message receivers (servers) are not required to provide a corresponding ASC X12 response in the timeframes specified in the transaction's corresponding Phase IV CAQH CORE Infrastructure Rule for Real Time.

For Batch submissions, the response must be only the standard SOAP message indicating whether the request was accepted or rejected. Message receivers (servers) must not respond to a Batch submission with an ASC X12 response, such as an ASC X12C v5010 999 in the HTTP response to the batch request, even if their systems' capabilities allow such a response. All ASC X12 responses must be available for pick up by the message sender (client) in accordance with the respective Phase IV CAQH CORE Infrastructure Rule for the transaction.

[29] MTOM is defined in Appendix §7.2: *Definitions and Abbreviations used in this Rule.*

4.2.5 Batch Response Pickup

Batch responses must be picked up after the message receiver (server) has had a chance to process a Batch submission corresponding ASC X12 response in the timeframes specified in the transaction's corresponding Phase IV CAQH CORE Infrastructure Rule.

Under this usage pattern, the message sender (client) connects to the message receiver (server) using HTTP/S and sends a SOAP message requesting available files, and the responder then sends back the file(s) in the HTTP/S SOAP response message (payload).

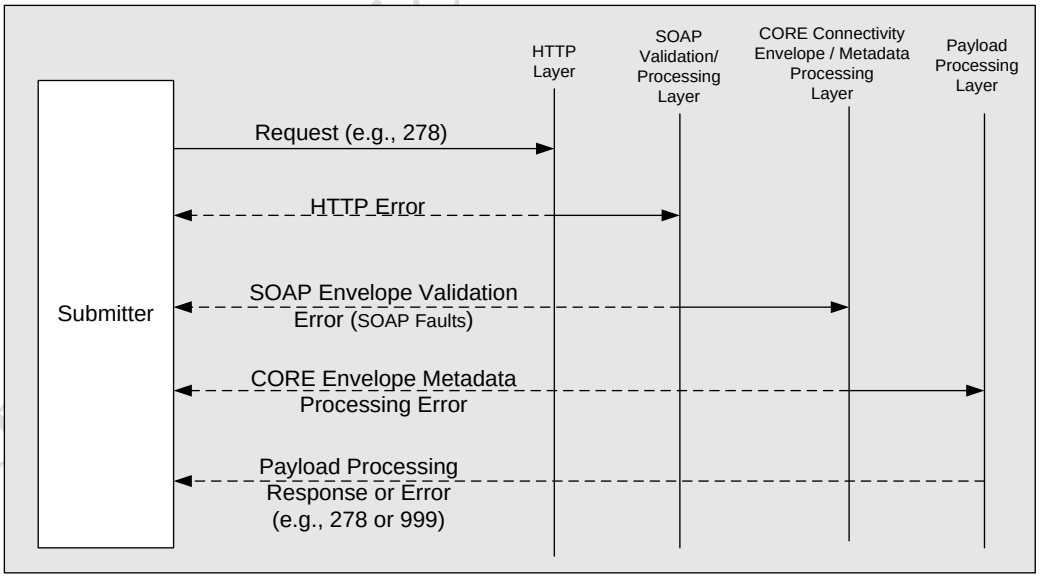
4.2.6 Error Handling

As shown in Figure #4.2.6 below, a submitted request goes through at least four logical layers that process the request. Errors relative to OSI Layers 3 and below are not addressed.

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

Once a request (e.g., ASC X12N v5010 278 Request) is submitted it goes through these 4 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 X12N v5010 278 Response or ASC X12C v5010 999 or ASC X12C TA1).

Figure #4.2.6



Note: In Figure #4.2.6 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.

4.2.6.1 HTTP Status and Error Codes (Normative, Not Comprehensive[30])

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 4.2.6.1 and are consistent with the HTTP status codes from Phase I CAQH CORE.

The status and error codes included in Table 4.2.6.1 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]. This rule requires the use of the appropriate HTTP error or status codes as applicable to the error/status situation. The list of status/error codes below is not intended to constrain the use of standard HTTP status/error codes relative to their original specification. The descriptions below are not intended to override the original definitions but to provide contextual information based on the use of these HTTP Status and Error Codes for CORE Connectivity error handling.

Table 4.2.6.1

Table 4.2.6.1	
HTTP Status/Error Codes (Normative, Not Comprehensive)	**CORE Rule Specific Description [31] (Intended Use)**
200 OK	Success
202 Accepted	Real Time or Batch file submission has been accepted (but not necessarily processed)
400 Bad Request	Incorrectly formatted HTTP headers
403 Forbidden	Access denied
500 Internal Server Error	The web-server encountered a processing error or there was a SOAP fault
5xx Server errors	Standard set of server side errors (e.g., 503 Service Unavailable)

4.2.6.2 SOAP Envelope Validation – SOAP Faults (Normative)

Errors at the SOAP Envelope validation layer are returned as SOAP faults [*http://www.w3.org/TR/soap12-part1/#soapfault*]. The full list of enumerated SOAP Faults may be found in the SOAP 1.2 specification. Table 4.2.6.2 provides perspective on two of the errors that are commonly used in relation to the CORE Rule.

The set of SOAP Faults below is not comprehensive – additional SOAP Faults that comply with the SOAP 1.2 specifications can be used. The descriptions below are not intended to override the original definitions but to provide contextual information based on the use of these SOAP Faults for CORE Connectivity error handling.

[30] 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].
[31] Section 6.1.1 of the HTTP specification http://tools.ietf.org/html/rfc2616#section-6.1.1.

737

Table 4.2.6.2	
SOAP Faults (Normative; Not Comprehensive)	**CORE Rule Specific Description (Intended Use)**
Sender	The envelope sent by the sender (client) 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.
Receiver	The message could not be processed for reasons attributable to the receiver (server) (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.

738

739 *4.2.6.3 CAQH CORE Connectivity Envelope Metadata Processing Status and Error Codes (Normative,*
740 *Comprehensive)*

741 To handle CORE-compliant envelope processing status and error codes, two fields called *ErrorCode* and
742 *ErrorMessage* are included in the CORE-compliant Envelope. (See §4.4.2) *ErrorMessage* is a free form text field
743 that describes the error (for the purpose of troubleshooting/logging). When an error occurs, *PayloadType* is set to
744 *CoreEnvelopeError*. The set of *ErrorCode*s in this table is normative and comprehensive, which means the use of
745 other error codes is not permitted.

746

Table 4.2.6.3	
CORE-compliant Envelope Processing Status/Error Codes (Normative, Comprehensive)	**CORE Status Code Description[32] (Intended Use)**
Success	Envelope was processed successfully.
<FieldName>Illegal	Illegal value provided for <FieldName>. Value provided is not valid based on the metadata constraints defined in the CORE Connectivity Rule.
<FieldName>UnSupported	Value is a legal value, but is not supported by the end point receiving the request. Server Connectivity Guide should indicate where to find specific SOAP Operations if multiple URLs are used to support CAQH CORE Phase IV Connectivity.
VersionMismatch	The CORE Rule Version sent is not valid at the receiver (server).
UnAuthorized	The sender could not be authorized (e.g., using the fields in the metadata, or using the client certificate information).
NotSupported	A request was received at this server with a valid *PayloadType* or *ProcessingMode* but is currently not implemented by this server (e.g., it may be implemented at a different server within this organization)

[32] 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].

Table 4.2.6.3	
CORE-compliant Envelope Processing Status/Error Codes (Normative, Comprehensive)	**CORE Status Code Description[32] (Intended Use)**
ChecksumMismatched	The checksum value computed on the recipient did not match the value that was sent in the envelope.

4.2.6.4 Examples of HTTP Status and Error Codes (non-normative)

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

- A SOAP request that has illegal HTTP headers gets a response with HTTP Error Code: "400 Bad Request."
- A SOAP request with an unauthenticated submitter's client-certificate 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.

4.2.6.5 Examples of SOAP Faults (non-normative)

The following illustrates some situations where "Sender" SOAP Faults may be returned:

- Invalid version of SOAP (e.g., SOAP 1.1)
- SOAP envelope does not have a SOAP Body
- SOAP Body does not contain the CAQH CORE Connectivity Elements

The following illustrates some situations where "Receiver" SOAP Faults may be returned:

- Failure to connect to a backend system for processing of the message

4.2.6.6 Examples of CORE Connectivity Envelope Metadata Processing 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=UnAuthorized*, the *ErrorMessage* could be "Unauthorized Sender – please contact XXX to get proper credentials".

- For *ErrorCode=NotSupported*, the *ErrorMessage* could be "The requested PayloadType is supported at a different URL, please review Connectivity Companion Guide"

4.2.7 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.

4.2.8 Tracking of Date and Time and Payload ID

In order to comply with the corresponding transaction's Phase IV CAQH CORE Infrastructure Rules message receivers (servers) 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 (clients) must include the date and time the message was sent in the CORE metadata element Time Stamp

4.2.9 Capacity Plan

4.2.9.1 Real Time Transactions

A HIPAA covered entity or its agent 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 (client) within response time requirements specified in the transaction's corresponding Phase IV CAQH CORE Infrastructure Rule.

Three major factors affect the specific number of Large Volume of Single Real Time Transactions (See §7.2) 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.

HIPAA covered entities or their agents must attest that their capacity planning addresses the above 3 factors that affect large volume single Real Time processing[33]. HIPAA covered entities or their agents 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.

4.2.9.2 Batch Transactions

The HIPAA covered entity or its agent's 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 (client) within the time specified in the applicable CORE Rule.

[33] See *Appendix 7.2: Abbreviations and Definitions used in this Rule* for a definition of Large Volume of Single Real time Transactions (Synchronous).

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.

HIPAA covered entities or their agents 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.

4.2.10 Real Time Response, Timeout and Retransmission Requirements

Real Time response time must conform to the transaction's corresponding Phase IV CAQH CORE Infrastructure Rule requirements.

- If a Real Time response message is not received within the 60 second response period, the submitter's (client) 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 (client) 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 (client) 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 Publication of Entity-Specific Connectivity Companion Document

Servers must publish detailed specifications in a Connectivity Companion Document on the entity's public web site. CORE recommends specifying the following. This list of recommendations is not intended to be either exhaustive or prohibitive as the specific details of a trading partner relationship are outside the scope of the CORE rules.

- 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).
- 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 payload for Batch processing mode that can be received by a Server.
- Authentication/Authorization policies using X.509 Client Certificates (e.g., how to enroll and obtain a Client Certificate to connect to that receiver (server).

- Information on obtaining the receiver's (server's) Root Certificate Authority and/or Intermediate Certificate Authority public key certificate.
- System Availability as required by the corresponding transaction's Phase IV CAQH CORE Infrastructure Rule.
- 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).
- If the Server only accepts FIPS 140-2 compliant connections, or if the Server organization security policy requires a stronger transport security than SSL v3.0, the version of TLS (1.1 or higher), and the algorithm (e.g., SHA-2) that is expected for Checksum element.

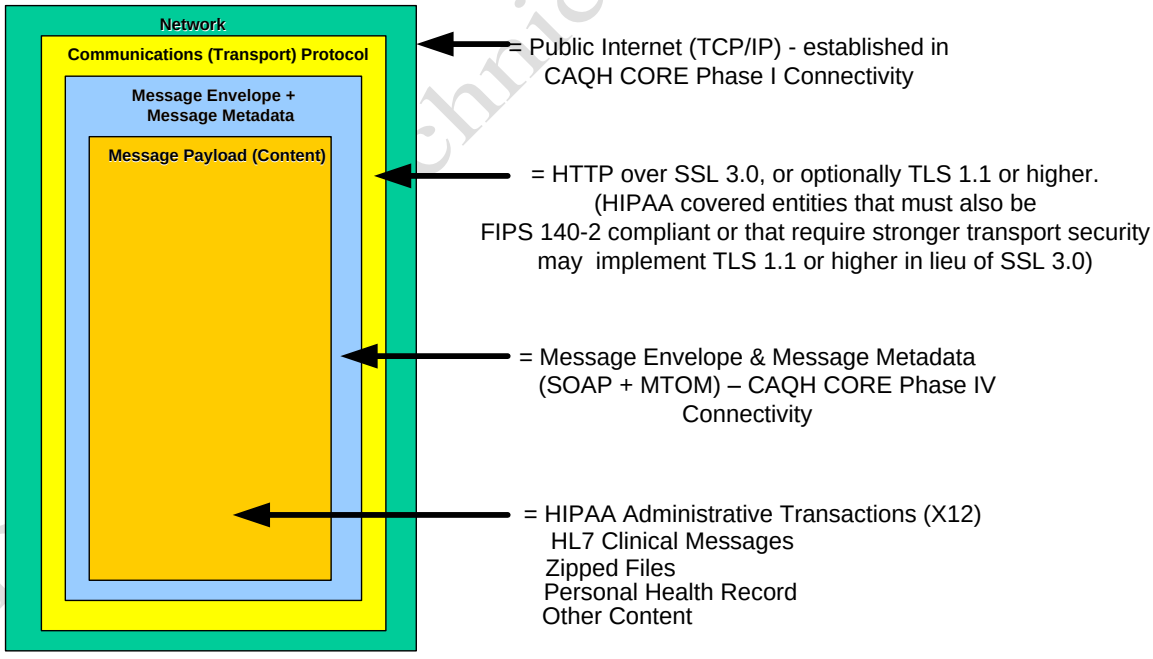
4.4 Envelope Metadata Fields, Descriptions, Intended Use and Syntax/Value-Sets

The Envelope Metadata specified in Table 4.4.2 below pertains to the Phase IV Message Envelope SOAP+WSDL. With the exception of *ErrorCode* and *ErrorMessage* fields, which are only sent in the response, the CORE Phase IV 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 CAQH CORE 153 Connectivity Rule version 1.1.0 established 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 #4.4.1



873 ### *4.4.2 Table of CORE Envelope Metadata*

874

Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[34]**	**Requirement Indicator for Real Time and Batch**	**DataType**	**Field Constraints or Value-sets (Not comprehensive)**
Payload Type	Payload Type specifies the type of payload included within a request, (e.g. HIPAA ASC X12N transaction set 837, 820, 278, etc.).	• Message routing • Efficient processing • Auditing	PayloadType	Required for both	Coded Set	Please see Phase IV CAQH CORE-required Payload Type Table document for enumeration of PayloadType field.
Processing Mode	Processing Mode indicates Batch or Real Time[35] 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[36] 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[37], to ensure uniqueness.

[34] Mixed case or Camel Case (e.g., *PayloadType*) capitalization is used for the field names to provide readability within the messages http://en.wikipedia.org/wiki/CamelCase

[35] See *Appendix 7.2: Abbreviations and Definitions used in this Rule* for a definition of Batch and Real time.

[36] 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.

[37] 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. However, the use of MAC address is not a requirement of this rule.

Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[34]**	**Requirement Indicator for Real Time and Batch**	**DataType**	**Field Constraints or Value-sets (Not comprehensive)**
Time Stamp	The Sender (request) or Receiver (response) Time Stamp 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)
Sender Identifier	A unique[38] 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[39] 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[40] 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. For response messages returned by a Server, this is the version of the Server implementation.	• Message routing and processing. • Auditing	CORERuleVersion	Required for both	Coded Set	4.0.0

[38] Unique within the Sender's (client's) domain.
[39] §2 shows the layers in the OSI model.
[40] Unique within a Receiver's (server's) domain.

Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[34]**	**Requirement Indicator for Real Time and Batch**	**DataType**	**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[41] Not used for Real Time	String	Algorithm is SHA-1[42] 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.

4.4.3 Specification of Processing Mode and Enumeration Payload Type Fields

4.4.3.1 Processing Mode Table (Normative)

A HIPAA covered entity or its agent must support the transaction processing mode requirements (i.e., Real Time and/or Batch) as specified in the *COREProcessingModePayloadTypeTables.docx* companion document to this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 when exchanging transactions in conformance with this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0.

The Processing Mode requirements specified in the Phase IV CAQH CORE-required Processing Mode Table also apply when a HIPAA covered entity or its agent are exchanging the transactions addressed by this rule using any other connectivity method as permitted by the CAQH CORE Safe Harbor. (See §5.)

4.4.3.2 Enumeration of Payload Types When Handling ASC X12 Payloads (Normative)

A HIPAA covered entity or its agent must support the requirements for identifying the payload (*PayloadType),* which is the essential data being carried within the content of the Message Envelope as specified in the

[41] 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.
[42] Entities requiring FIPS 140-2 compliance may use SHA-2 instead of SHA-1. If SHA-2 is used, then the entity's Connectivity Companion Document will specify that SHA-2 is expected in incoming messages from trading partners.

COREProcessingModePayloadTypeTables.docx companion document to this Phase IV CORE Connectivity Rule. (See Figure #4.4.1, Table 4.4.2, and §6.1.). (See §3.7.3 for maintenance requirements.)

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

The Envelope metadata specification in §4.4.3 includes a PayloadType field that is enumerated for ASC 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 HL7 based ADT04 Version 2.3.1 payload may specify the *PayloadType* as *HL7_ADT04_2_3_1*.

5 CORE SAFE HARBOR

This rule specifies a "Safe Harbor" that any stakeholder can be assured will be supported by any HIPAA covered entity or its agent. This rule further specifies the connectivity method that all HIPAA covered entities or their agents and all voluntarily CORE-certified organizations must implement and with which conformance must be demonstrated.

As such, this 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 trading partners to use a CORE-compliant method for all new connections.
- **DOES NOT** require all trading partners to use only one method for any connections.
- **DOES NOT** require any entity to do business with any trading partner or other entity.

CAQH CORE expects that in some circumstances, trading partners may agree to use different communication method(s) and/or security requirements than those described in this rule to achieve the technical goals of the specific connection. Examples of potential different communication methods that could be implemented under this CAQH CORE Safe Harbor provision include a VPN (virtual private network) or SFTP (secure file transfer protocol.) Such connectivity gateways are not considered compliant with this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0. When a HIPAA covered entity or its agent implements a different communication method(s) as permitted by this CAQH CORE Safe Harbor all payload processing modes specified for the transactions addressed by this rule must be supported in each connectivity gateway implemented which does not comply with this Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 requirements. (See §4.4.3.1)

This Phase IV CAQH CORE 470 Connectivity Rule v4.0.0 is the CAQH CORE Safe Harbor connectivity method that a HIPAA covered entity or its agent MUST use if requested by a trading partner. If the HIPAA covered entity or its agent do not believe that this CAQH CORE Safe Harbor is the best connectivity method 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 CAQH CORE Safe Harbor, the HIPAA covered entity or its agent 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 CONFORMANCE REQUIREMENTS

Conformance with this CAQH CORE Operating Rule can be voluntarily demonstrated and certified through successful completion of the approved CORE test suite with a third party CORE-authorized testing vendor,

followed by the entity's successful application for a CORE Certification. A CORE Certification demonstrates that an entity has successfully tested for conformity with all of the Phase IV CAQH CORE Operating Rules, and the entity or its product has fulfilled all relevant conformance requirements.

Only the Department of Health and Human Services (HHS) can decide whether a particular entity's system is **compliant** or **noncompliant** with HIPAA Administrative Simplification requirements (which include HIPAA-adopted CAQH CORE Operating Rules). HHS may adjudicate on an entity's compliance and assess civil money penalties or penalty fees for noncompliance under the following HIPAA Administrative Simplification mandates:

- HIPAA regulations mandate that the Secretary "will impose a civil money penalty upon a covered entity or business associate if the Secretary determines that the covered entity or business associate has violated an administrative simplification provision." (47 CFR 160.402) Under the Affordable Care Act, HIPAA mandates a certification process for health plans only, under which health plans are required to file a statement with HHS certifying that their data and information systems are in compliance with applicable standards and associated operating rules. (Social Security Act, Title XI, Section 1173(h)). HIPAA also mandates that a health plan must "ensure that any entities that provide services pursuant to a contact with such health plan shall comply with any applicable certification and compliance requirements"(Social Security Act, Title XI, Section 1173(h)(3)).
- HIPAA also mandates that HHS is to "conduct periodic audits to ensure that health plans… are in compliance with any standards and operating rules." (Social Security Act, Title XI, Section 1173(h))

7 APPENDIX

7.1 References

Note: These were used for Rule creation as well as to create the analysis artifacts as part of CORE Phase IV 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
NIST 800-52r1	Guidelines for the Selection, Configuration and Use of Transport Layer Security (TLS) Implementations	http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-52r1.pdf
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

7.2 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)[43] 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[44] 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 both the requester and responder. 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 ASC X12 Interchange containing one Functional Group containing one ASC X12 transaction set consisting of more than one business transaction.
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 HIPAA covered entities or their agents.

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

Term or Concept	Definition
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.[45] 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.[46]
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).
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.[47] 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[48].

[45] http://www.atis.org/glossary/definition.aspx?id=7853 ATIS (Alliance for Telecommunications Industry Solutions http://www.atis.org/about/index.asp
[46] http://en.wikipedia.org/wiki/Extensibility
[47] Adapted from http://engineers.ihs.com/document/abstract/AQSBFBAAAAAAAAAA ANSI Information Technology – Vocabulary – Part 1: Fundamental Terms
[48] HIMSS Position Statement: Adoption of HITSP Interoperability Specifications July 2007

Term or Concept	Definition
Interoperability Specification[49]	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.
Media Access Control (MAC) Address	A media access control address (MAC address) is a unique identifier assigned to network interfaces for communications on the physical network segment.
Message Encapsulation Layer	This refers to the Open Systems Interconnect (OSI) layers 5 and 6.
Message Envelope Standard	SOAP+WSDL, described in Section "Specifications for SOAP + WSDL".
Metadata	Data about data. In the context of CORE Connectivity, metadata is the information in the message envelope that describes the payload.
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].

[49] 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 [50]	"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).[51]
Performance	According to the Phase I CAQH CORE 153 Connectivity Rule, performance is defined in only two components: Response time – the time required to receive a Request, process it completely and send an appropriate response, as specified in the Phase I CAQH CORE Eligibility and Benefits Rules and Policies for Real time [52] and Batch [53] exchanges. System Availability – the time an information source's (health plan, clearinghouse/switch or other intermediary system) processing system is capable of properly processing Request/Response transactions, as specified in the Phase I CAQH CORE Eligibility and Benefits Rules and Policies for system availability [54].

[50] International Telecommunication Union – Open Standards Definition. http://www.itu.int/ITU-T/othergroups/ipr-adhoc/openstandards.html
[51] SearchSecurity.com. http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci214475,00.html
[52] CORE Phase I 156: Eligibility and Benefits Real Time Response Time Rule
[53] CORE Phase I 155: Eligibility and Benefits Batch Response Time Rule
[54] CORE Phase I 157: Eligibility and Benefits System Availability

Term or Concept	Definition
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 a Request, process it completely, and send an appropriate response.[55] Maximum Arrival Rate Before Saturation – the maximum number of properly formed arriving Request transactions per time period (usually seconds or minutes), above which the ability for increased acceptance for further processing stops.[56] 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 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.[57] Capacity – the maximum number of completed Request/ Response transaction sets per specific time period. Quality of Service – the number of properly and accurately completed 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.
Processing Mode	Processing modes or computing modes are classifications of different types of computer processing, e.g., batch, real time. In the context of CAQH CORE Operating rules, the concept of processing mode applies to the timeframe within which a receiver of a payload of transactions processes those transactions and returns to the sender of the payload appropriate acknowledgements. See Batch and Real Time for CAQH CORE definitions.
Real time (Real time Mode, Real time Processing Mode) [58]	Real time Mode [59] is when an entity is required to send a transaction and receive a 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 both the requester and responder.

[55] CORE Phase I 156: Eligibility and Benefits Real Time Response Time Rule; and CORE Phase I 155: Eligibility and Benefits Batch Response Time Rule
[56] http://www.cs.washington.edu/homes/lazowska/qsp/Contents.pdf Quantitative System Performance, Chapter 5.2.1. Transaction Workloads (Page 72)
[57] http://www.atis.org/tg2k/ and search "Overhead Information" ATIS (Alliance for Telecommunications Industry Solutions http://www.atis.org/about.shtml
[58] CORE Phase I Glossary Definitions. www.caqh.org/pdf/COREPIGlossary.pdf
[59] CORE Phase I Glossary Definitions. http://www.caqh.org/pdf/COREPIGlossary.pdf

Term or Concept	Definition
Safe Harbor	A "Safe Harbor" is generally defined as a statutory or regulatory provision that provides protection from a penalty or liability.[60] 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.[61]
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.
Submitter Authentication	X.509 Certificate based Authentication over SSL or TLS, 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)[62] 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). Future enhancements and development by the IETF will occur within the TLS specification.
WSDL	W3C Web Services Definition Language Version 1.1 (http://www.w3.org/TR/2001/NOTE-wsdl-20010315).

952

[60] Merriam-Webster's Dictionary of Law. Merriam-Webster, Inc., 28 May, 2007. <Dictionary.com http://dictionary.reference.com/browse/safeharbor>
[61] http://isotc.iso.org/livelink/livelink/fetch/2000/2122/830949/3934883/3935096/07_gen_info/faq.html
[62] http://en.wikipedia.org/wiki/Transport_Layer_Security

7.3 Sequence Diagrams

The UML sequence diagrams below show interactions between a client and a server. 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 ASC X12C 999 or ASC X12C TA1 payloads).

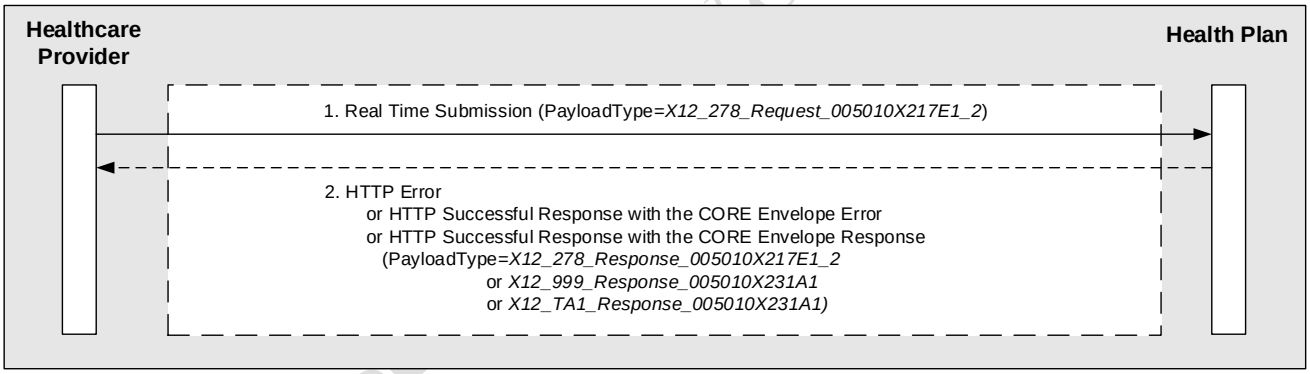
7.3.1 Real Time Interaction

This section describes Real Time interactions that include the following steps:

- Submission of Real Time Payload (step 1 in the diagrams)
- Real Time (Synchronous) response (step 2 in the diagrams)

Example 1: Health Care Services Review – Request for Review and Response (ASC X12N v5010 278)

The UML sequence diagram below shows a *Health Care Services Review – Request for Review and Response* Real Time transaction between a HIPAA covered Healthcare Provider and a HIPAA covered Health Plan. The interactions are described in the diagram below.



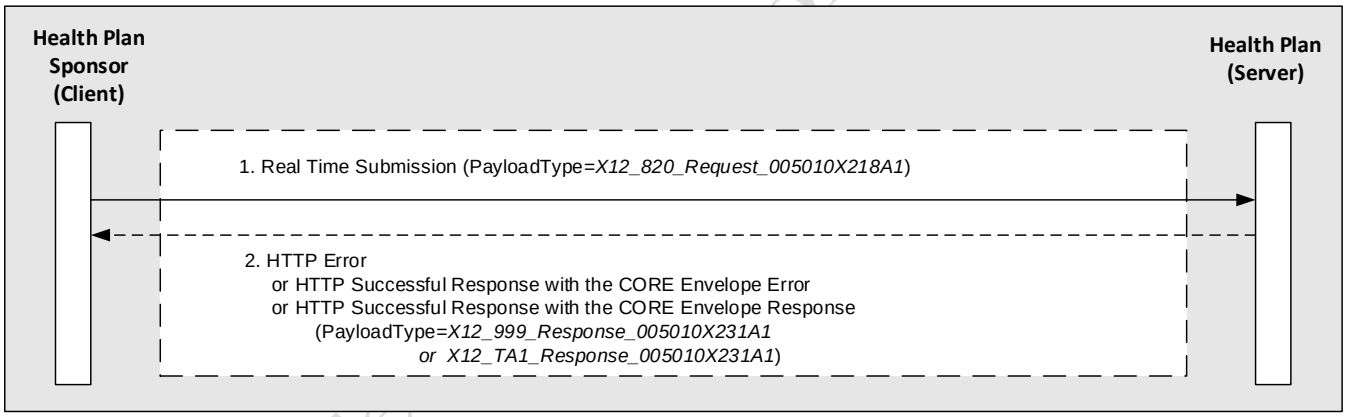
The requester of a Real Time response expects one and only one response on the payload; for example, in the above message interaction, the response payload can only be an ASC X12N v5010 278, or an ASC X12C v5010 999 or an ASC X12C TA1. The following describes the Real Time interaction as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Healthcare Provider submits a Real Time request to a Health Plan, using PayloadType=*X12_278_Request_005010X217E1_2*.	Health Care Services Review - Request for Review & Response

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
2	Health Plan responds (synchronously to request message 1) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_278_Response_005010X217E1_2* or *X12_999_Response_005010X231A1* or *X12_TA1_Response_005010X231A1)*	Health Care Services Review - Request for Review & Response

Example 2: Payroll Deducted and Other Group Premium Payment for Insurance Products (ASC X12N v5010 820)

The UML sequence diagram below shows a *Payroll Deducted and Other Group Premium Payment for Insurance Products* Real Time transaction between a HIPAA covered Health Plan Sponsor (Client) and a HIPAA covered Health Plan (Server). The interactions are described in the diagram below.



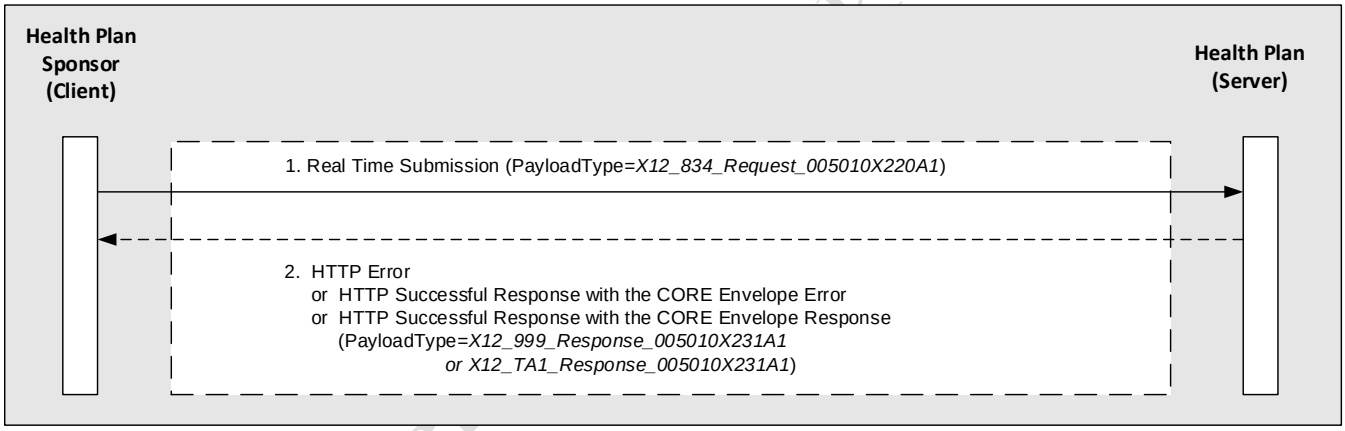
The requester of a Real Time response expects one and only one response on the payload; for example, in the above message interaction, the response payload can only be an ASC X12C v5010 999, or an ASC X12C TA1. The following describes the Real Time interaction as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Health Plan Sponsor submits a Real Time request to a Health Plan, using PayloadType=*X12_820_Request_005010X218A1*.	Payroll Deducted and Other Group Premium Payment for Insurance Products

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
2	A Health Plan responds (synchronously to request message 1) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_999_Response_005010X231A1* *or X12_TA1_Response_005010X231A1*)	Payroll Deducted and Other Group Premium Payment for Insurance Products

Example 3: Benefit Enrollment and Maintenance (ASC X12N v5010 834)

The UML sequence diagram below shows a *Benefit Enrollment and Maintenance* Real Time transaction between a Health Plan Sponsor (Client) and a Health Plan (Server). The interactions are described in the diagram below.



The requester of a Real Time response expects one and only one response on the payload; for example, in the above message interaction, the response payload can only be only a ASC X12C v5010 999, or a ASC X12C TA1. The following describes the Real Time interaction as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Health Plan Sponsor submits a Real Time request to a Health Plan, using PayloadType=*X12_834_Request_005010X218A1*.	Benefit Enrollment and Maintenance

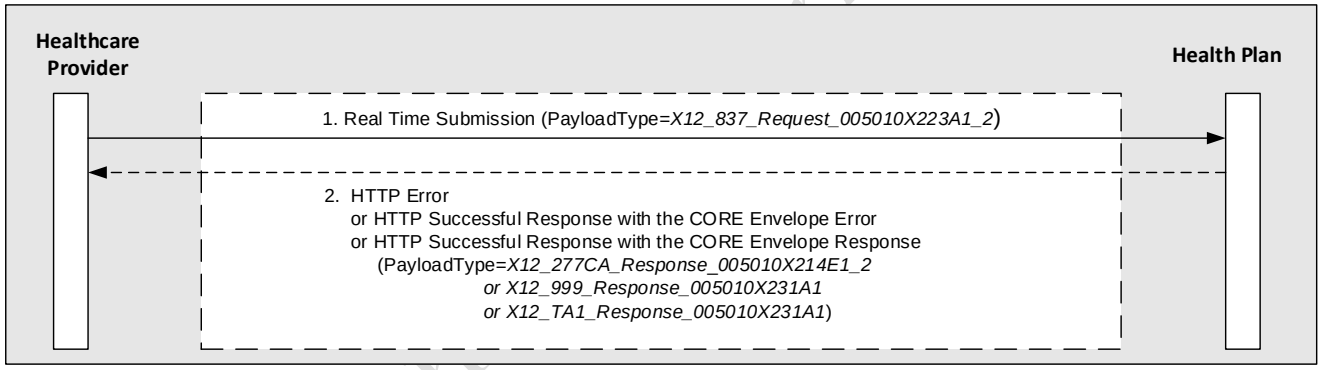
Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
2	A Health Plan responds (synchronously to request message 1) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_999_Response_005010X231A1* *or X12_TA1_Response_005010X231A1*)	Benefit Enrollment and Maintenance

995

996 *Example 4: Healthcare Claim (ASC X12N v5010 837 Claim)*

997 The UML sequence diagram below shows an Institutional *Healthcare Claim* Real Time transaction between a
998 HIPAA covered Healthcare Provider (Client) and a HIPAA covered Health Plan (Server). The interactions are
999 described in the diagram below.

1000



1001

1002 The requester of a Real Time response expects one and only one response on the payload; for example, in the
1003 above message interaction, the response payload can only be an ASC X12N v5010 277CA, or an ASC X12C
1004 v5010 999, or an ASC X12C TA1. The following describes the Real Time interaction as shown in the above
1005 diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Health Plan Sponsor submits a Real Time request to a Health Plan, using PayloadType=*X12_837_Request_005010X223A1_2*.	Healthcare Claim: Institutional

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
2	A Health Plan responds (synchronously to request message 1) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_277CA_Response_005010X214E1_2* *or X12_999_Response_005010X231A1* *or X12_TA1_Response_005010X231A1*)	Healthcare Claim: Institutional

7.3.2 Batch Interactions

This section describes Batch interactions that include the following steps:

- Submission of Batch Payload (steps 1 and 2 in the diagrams)
- Retrieval of Acknowledgment for the submission (steps 3 and 4 in the diagrams)
- Retrieval of Batch Processing Results (steps 5 and 6 in the diagrams)
- Submission of Acknowledgment for the results retrieved (steps 7 and 8 in the diagrams)

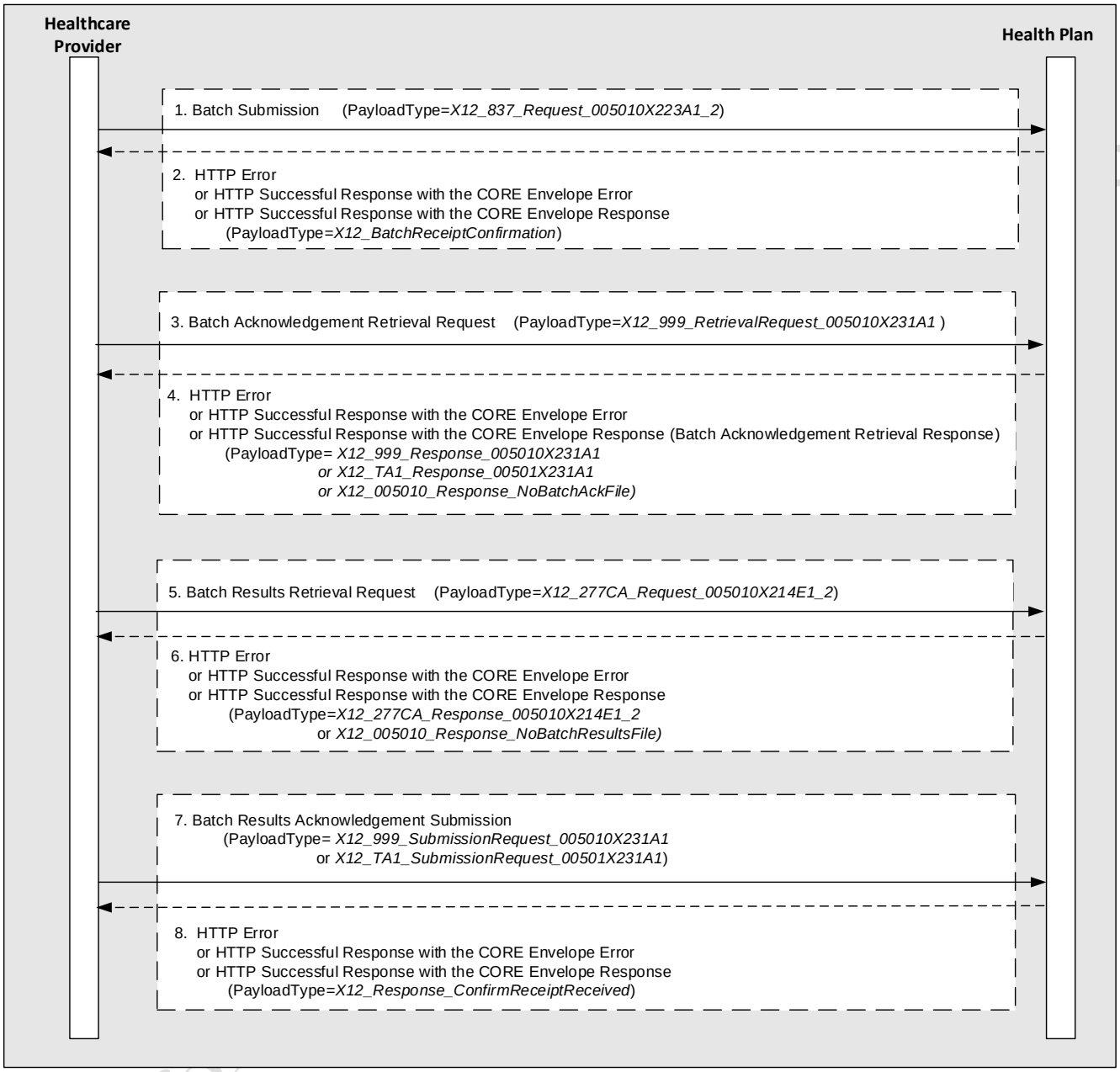
The Batch interactions can be conducted using specific payload types as shown in 7.3.2.1 or with Mixed Payload types as show in 7.3.2.2.

7.3.2.1 Batch Interaction for Specific Payload Types

Within the Batch Interaction for Specific Payload Types, the Batch Payload consists of a single type of transaction set.

Example 1: Health Care Claim (ASC X12N v5010 837 Claim):

The UML sequence diagram below shows a typical Batch Interaction between a HIPAA covered Healthcare Provider, and a HIPAA covered Health Plan specifically for ASC X12N v5010 837 batch payloads.



1021
1022

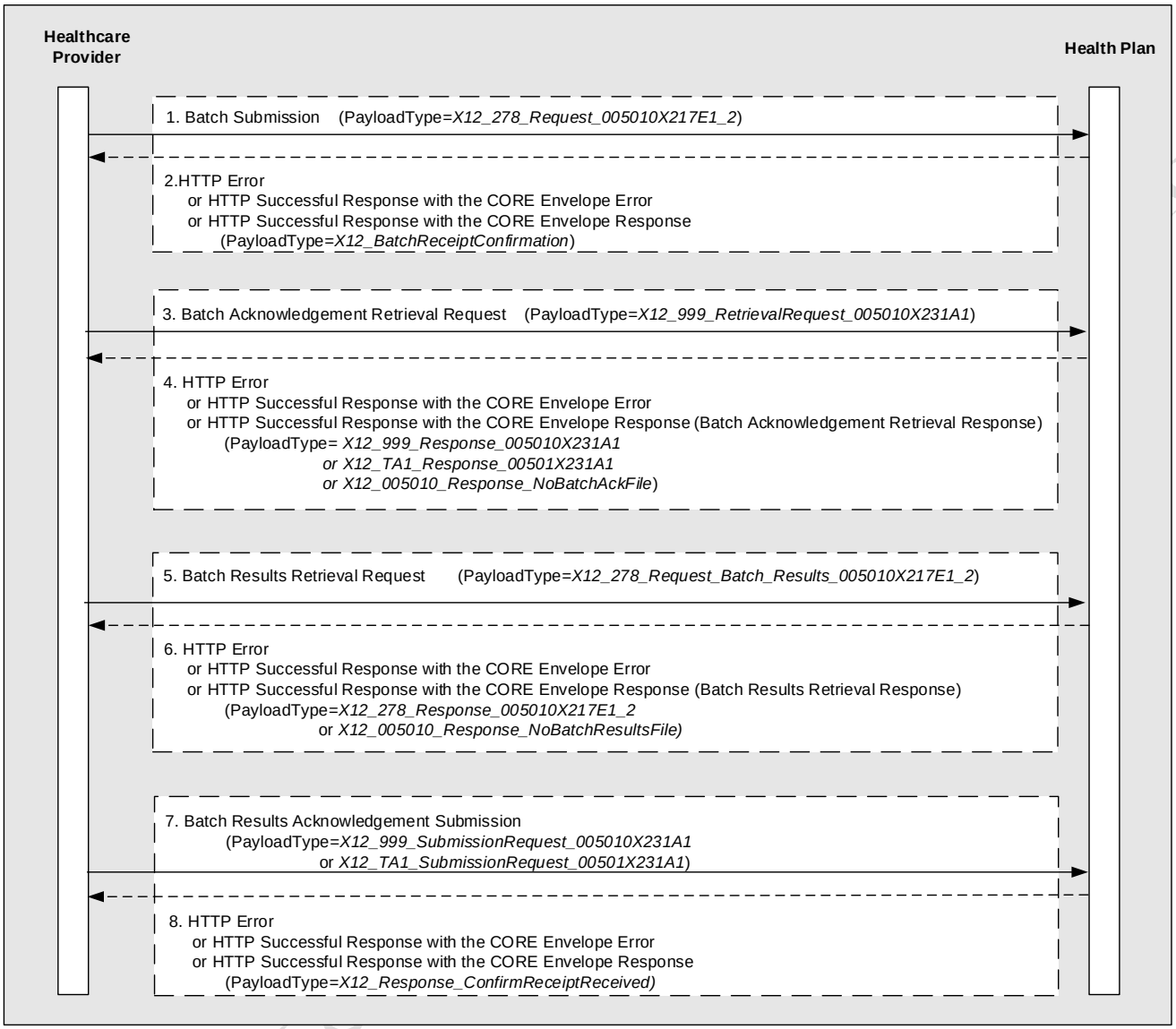
1023 The following describes the Batch interaction as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Healthcare Provider submits a Batch of requests to a Health Plan, using PayloadType = *X12_837_Request_005010X223A1_2 for an Institutional claim, or X12_837_Request_005010X222A1 for a Professional claim, or X12_837_Request_005010X224A1_2 for a Dental Claim.*	Health Care Claim: Institutional
2	A Health Plan responds (synchronously to request message 1) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_BatchReceiptConfirmation*)	Batch Receipt Confirmation Response
3	A Healthcare Provider sends a Request to a Health Plan to solicit the acknowledgement (*X12_999_RetrievalRequest_005010X231A1*) for the Batch file that was just submitted.	Implementation Acknowledgement Retrieval
4	A Health Plan responds (synchronously to request message 3) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (Batch Acknowledgement Retrieval Response) (PayloadType= *X12_999_Response_005010X231A1* *or X12_TA1_Response_00501X231A1* *or X12_005010_Response_NoBatchAckFile)*	Implementation Acknowledgement Retrieval
5	A Healthcare Provider sends a Request to a Health Plan to solicit the Health Care Claim Acknowledgement for the batch of claims that was submitted in message sequence 1 using PayloadType=*X12_277CA_Request_005010X214E1_2*.	Health Care Claim Acknowledgement
6	A Health Plan responds (synchronously to request message 5) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_277CA_Response_005010X214E1_2* or *X12_005010_Response_NoBatchResultsFile)*	Health Care Claim Acknowledgement

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
7	A Healthcare Provider submits the acknowledgement Batch Results Acknowledgement Submission (PayloadType= *X12_999_SubmissionRequest_005010X231A1* or *X12_TA1_SubmissionRequest_00501X231A1*) to a Health Plan. This acknowledgment submission is required by the Phase IV CAQH CORE Infrastructure Rule corresponding to the specific transaction.	Implementation Acknowledgement Submission (Request)
8	A Health Plan responds (synchronously to request message 7) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_Response_ConfirmReceiptReceived*)	Implementation Acknowledgement Submission (Response)

Example 2: Health Care Services Review – Request for Review & Response (ASC X12N v5010 278):

The UML sequence diagram below shows a typical Batch Interaction between a HIPAA covered Healthcare Provider and a HIPAA covered Health Plan for ASC X12N v5010 278 batch payloads.



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

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Healthcare Provider submits a Batch of requests to a Health Plan, using PayloadType as *X12_278_Request_005010X217E1_2*.	Health Care Services Review – Request for Review & Response

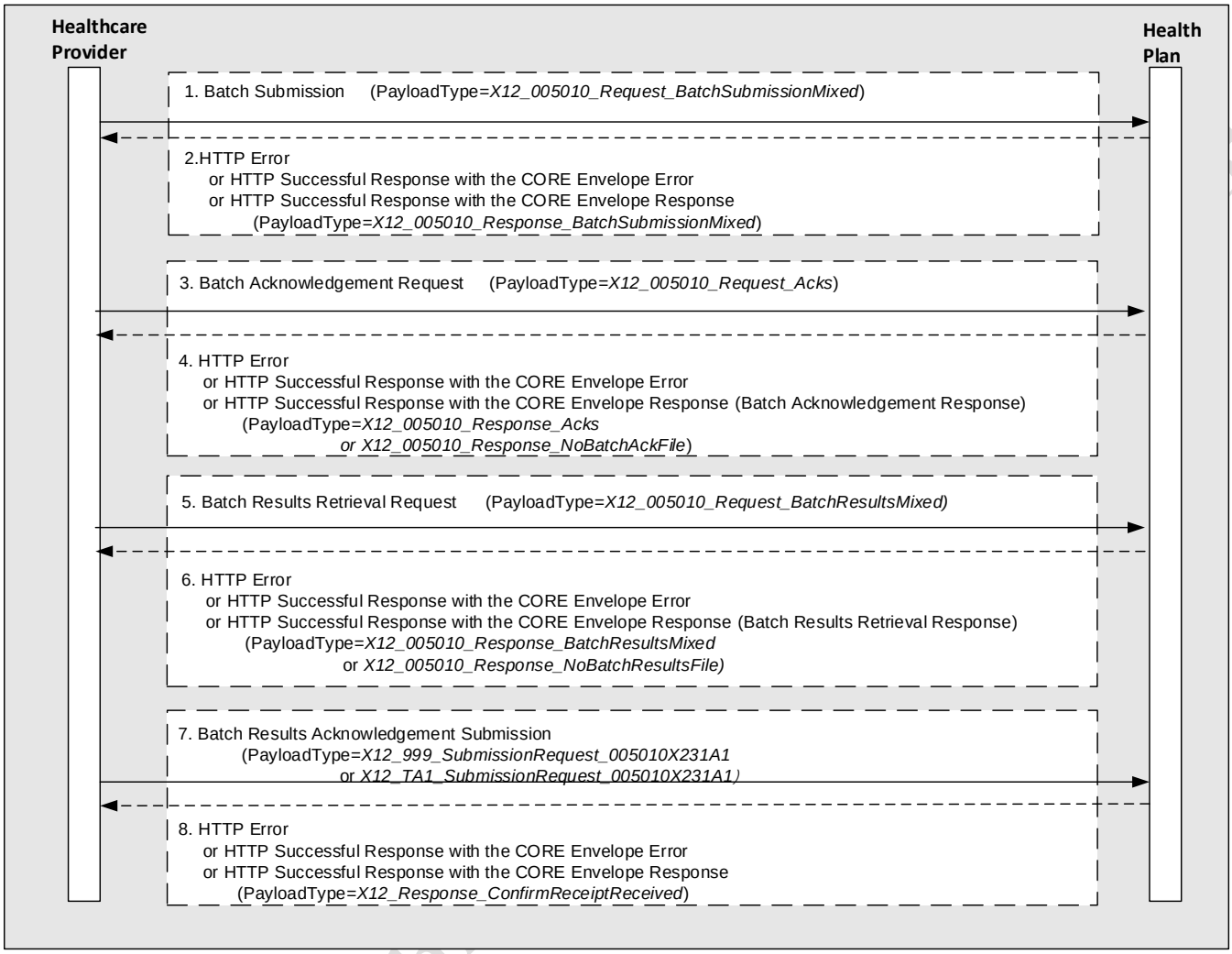
Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
2	A Health Plan responds (synchronously to request message 1) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_BatchReceiptConfirmation*)	Batch Receipt Confirmation Response
3	A Healthcare Provider sends a Request to a Health Plan to solicit the acknowledgement (*X12_999_RetrievalRequest_005010X231A1*) for the Batch file that was just submitted.	Implementation Acknowledgement Retrieval
4	A Health Plan responds (synchronously to request message 3) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (Batch Acknowledgement Retrieval Response) (PayloadType= *X12_999_Response_005010X231A1* *or X12_TA1_Response_00501X231A1* *or X12_005010_Response_NoBatchAckFile*)	Implementation Acknowledgement Retrieval
5	A Healthcare Provider sends a Request to a Health Plan to solicit the results of processing the batch that was submitted in message sequence 1, using Payload Type: *X12_278_Request_005010X217E1_2*.	Health Care Services Review – Request for Review & Response
6	A Health Plan responds (synchronously to request message 5) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (Batch Results Retrieval Response) (PayloadType=*X12_278_Response_005010X217E1_2* or *X12_005010_Response_NoBatchResultsFile)*	Health Care Services Review – Request for Review & Response (
7	A Healthcare Provider submits the acknowledgement (PayloadType= *X12_999_SubmissionRequest_005010X231A1*, or *X12_TA1_SubmissionRequest_00501X231A1)* to a Health Plan. This acknowledgment submission is required by CORE Phase I and Phase II Rules.	Implementation Acknowledgement Submission

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
8	A Health Plan responds (synchronously to request message 7) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_Response_ConfirmReceiptReceived)*	Implementation Acknowledgement Submission

1031

1032 *7.3.2.2 Batch Interaction for Mixed Payload Types*

1033 The UML sequence diagram below shows a Mixed Payload Type Batch Interaction between a HIPAA covered
1034 Healthcare Provider and a HIPAA covered Health Plan.



1035

1036 The following describes the typical Mixed Batch interaction as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Healthcare Provider submits a Batch of requests to a Health Plan, using PayloadType=*X12_005010_Request_BatchSubmissionMixed*)	Batch Submission (mixed payload types)
2	A Health Plan responds (synchronously to request message 1) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_005010_Response_BatchSubmissionMixed*)	Batch Submission (mixed payload types)
3	A Healthcare Provider sends a Request to a Health Plan with PayloadType=*X12_005010_Request_Acks* to solicit the acknowledgement from a Health Plan (ASC X12C v5010 999 or ASC X12C TA1) for the Batch file that was just submitted.	General Acknowledgements Pick Up
4	A Health Plan responds (synchronously to request message 3) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (Batch Acknowledgement Response) (PayloadType=*X12_005010_Response_Acks* *or X12_005010_Response_NoBatchAckFile*)	General Acknowledgements Pickup
5	A Healthcare Provider sends a Request to a Health Plan to solicit the Results for the Batch file that was submitted in message sequence 1 using PayloadType=X12_005010_Request_BatchResultsMixed.	Batch Results Retrieval (mixed payload types)
6	A Health Plan responds (synchronously to request message 5) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (Batch Results Retrieval Response) (PayloadType=*X12_005010_Response_BatchResultsMixed* or *X12_005010_Response_NoBatchResultsFile)*	Batch Results Retrieval (mixed payload types)
7	A Healthcare Provider submits the acknowledgement (PayloadType= *X12_999_SubmissionRequest_005010X231A1* or *X12_TA1_SubmissionRequest_005010X231A1)* to a Health Plan. This acknowledgment submission is required by CORE Phase I and Phase II Rules.	Implementation Acknowledgement Submission

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
8	A Health Plan responds (synchronously to request message 7) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_Response_ConfirmReceiptReceived*)	Implementation Acknowledgement Submission

7.3.3 Generic Batch Interactions

The term *Generic* is used to denote the fact that the Batch Interactions defined herein can be used as building blocks to build more complex interactions if such interactions are needed to support current or future business use cases. Within the Generic Batch Interactions, there are two types:

1) *Generic Push*: this message interaction is characterized by the following steps:
   - Client submits, or "pushes" a Batch Payload to a Server
   - Client then retrieves an acknowledgment (or error) from the Server for the Batch Payload that it had previously submitted to the Server.
2) *Generic Pull*: this message interaction is characterized by the following steps:
   - Client retrieves, or "pulls" a Batch Payload from a Server
   - Client then submits an acknowledgment (or error) to the Server for the Batch Payload that the Client has previously retrieved from the Server.

Both of these message interactions can be used either for Specific Transaction Batch Payload Types (with a single type of transaction set), or for Mixed Batch Payload types (using multiple transaction sets within the same Batch Payload). For simplicity, the examples shown below are limited to Specific Transaction Batch Payload Types.

Two example transactions are shown in the following sub-sections:

a) Benefit Enrollment and Maintenance (ASC X12N v5010 834)

b) Payroll Deducted and Other Group Premium Payment for Insurance Products (ASC X12N v5010 820)

Both of these transactions can use either the *Generic Push* or *Generic Pull* interactions. Depending on the interaction being used, the business actors that use these interactions will need to assume the roles of Client or Server.

7.3.3.1 Generic Push

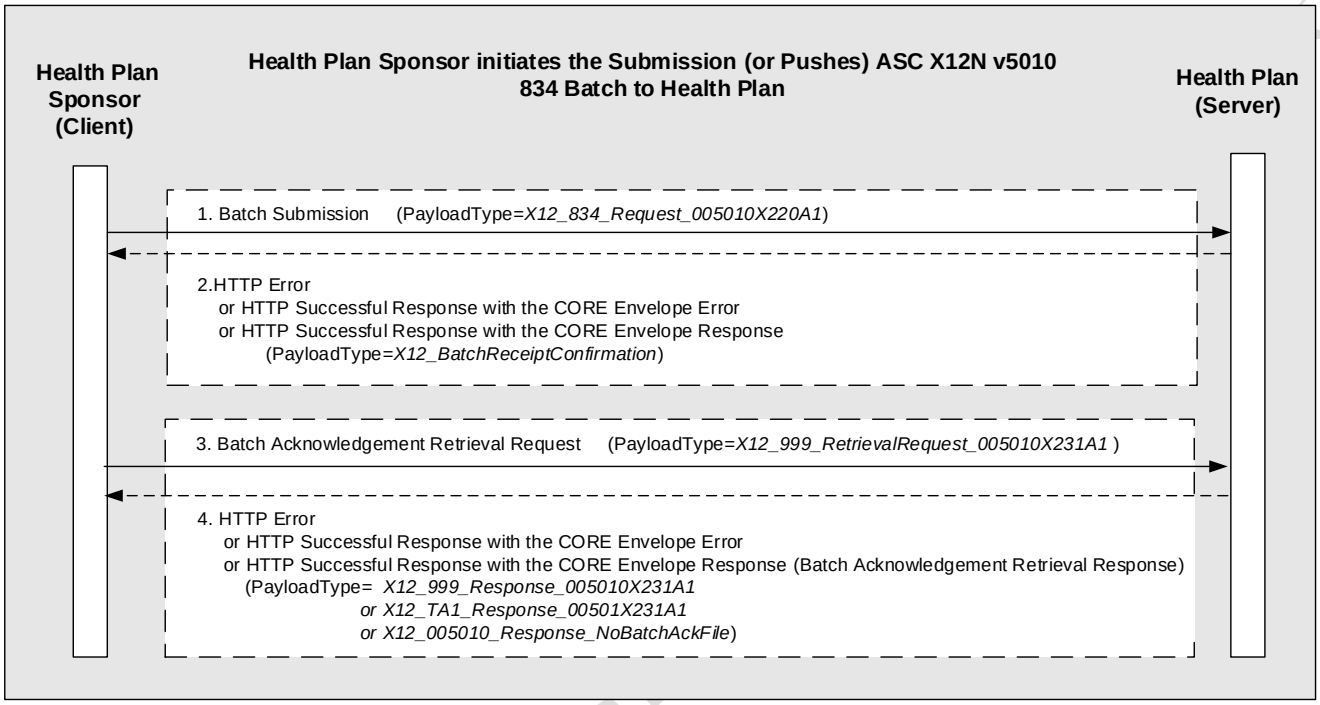
This message interaction is characterized by the following steps:

- Client submits, or "pushes" a Batch Payload to a Server
- Client then retrieves an acknowledgment (or error) from the Server for the Batch Payload that it had previously submitted to the Server.

1064 The UML sequence diagrams below show examples of the Generic Push Interactions.

1065
1066 *Example: Benefit Enrollment and Maintenance (ASC X12N v5010 834)*

1067



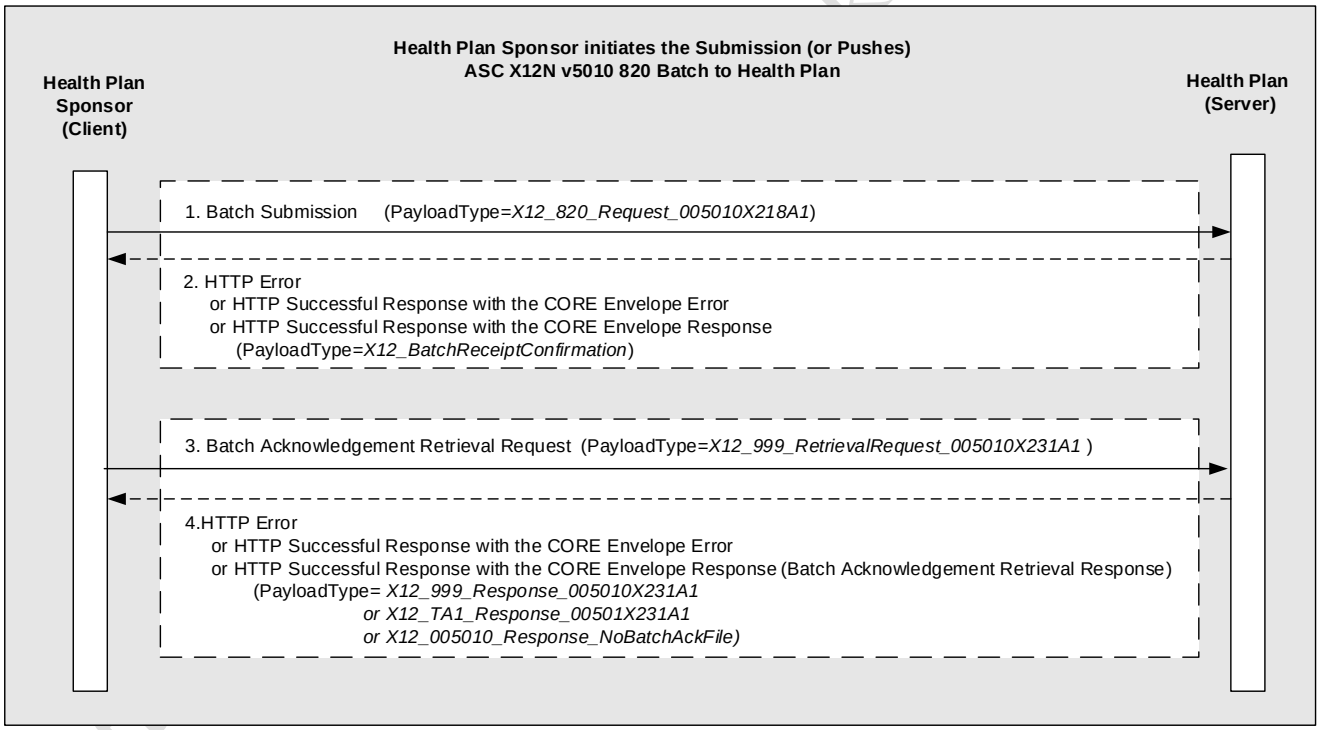
1068

1069 The following describes the *Benefit Enrollment and Maintenance* transaction using the *Generic Push* interaction
1070 as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Health Plan Sponsor (Client) submits to a Health Plan (Server) a Batch of Benefit Enrollment and Maintenance requests using PayloadType=*X12_834_Request_005010X220A1*.	Benefit Enrollment and Maintenance
2	A Health Plan (Server) responds synchronously in Real time either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_BatchReceiptConfirmation*)	Batch Receipt Confirmation Response
3	A Health Plan Sponsor (Client) sends a Request to a Health Plan (Server) with (PayloadType=*X12_999_RetrievalRequest_005010X231A1*) to solicit the acknowledgement (ASC X12C v5010 999 or ASC X12C TA1) for the Batch file that was just submitted.	Implementation Acknowledgement Retrieval

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
4	A Health Plan (Server) responds (synchronously to request message 3) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (Batch Acknowledgement Retrieval Response) (PayloadType= *X12_999_Response_005010X231A1* *or X12_TA1_Response_00501X231A1* *or* *X12_005010_Response_NoBatchAckFile*)	Benefit Enrollment and Maintenance

Example: Payroll Deducted and Other Group Premium Payment for Insurance Products (ASC X12N v5010 820)



The following describes the *Payroll Deducted and Other Group Premium Payment for Insurance Products* transaction using the *Generic Push* interaction, as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	The Health Plan Sponsor (Client) submits to a Health Plan (Server) a Batch of Payroll Deducted and Other Group Premium Payment for Insurance Products requests using PayloadType= *X12_820_Request_005010X218A1*.	Payroll Deducted and Other Group Premium Payment for Insurance Products
2	A Health Plan (Server) responds synchronously in Real time either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_BatchReceiptConfirmation*)	Payroll Deducted and Other Group Premium Payment for Insurance Products
3	A Health Plan Sponsor (Client) sends a Request to a Health Plan (Server) using (PayloadType=*X12_999_RetrievalRequest_005010X231A1*) to solicit the acknowledgement (ASC X12C v5010 999 or ASC X12C TA1) for the Batch file that was just submitted.	Implementation Acknowledgement Retrieval
4	A Health Plan (Server) responds (synchronously to request message 3) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (Batch Acknowledgement Retrieval Response) (PayloadType= *X12_999_Response_005010X231A1* *or X12_TA1_Response_00501X231A1* *or* *X12_005010_Response_NoBatchAckFile)*	Implementation Acknowledgement Retrieval (Response)

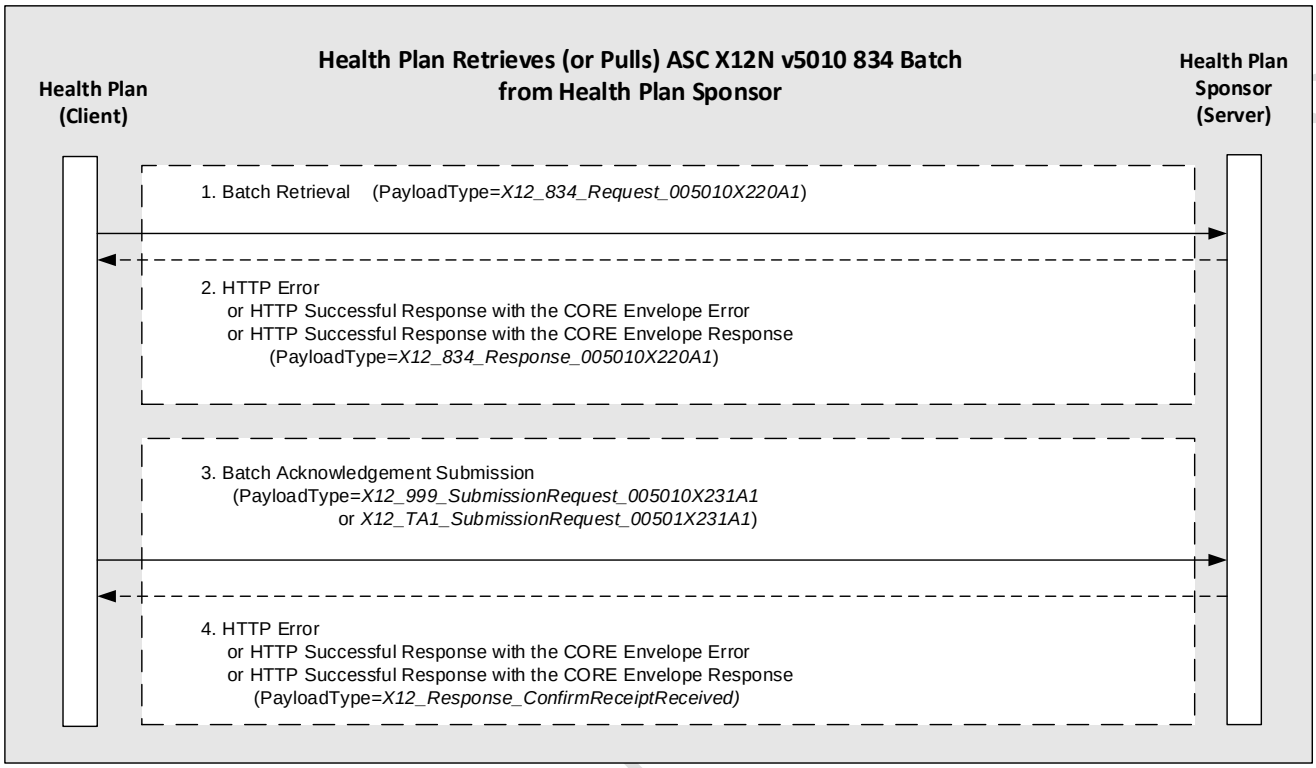
7.3.3.2 Generic Pull

This message interaction is characterized by the following steps:

- Client retrieves, or "pulls" a Batch Payload from a Server
- Client then submits an acknowledgment (or error) to the Server for the Batch Payload that the Client has previously retrieved from the Server.

The UML sequence diagrams below show examples of the *Generic Pull* Interactions.

Example: Benefit Enrollment and Maintenance (ASC X12N v5010 834)

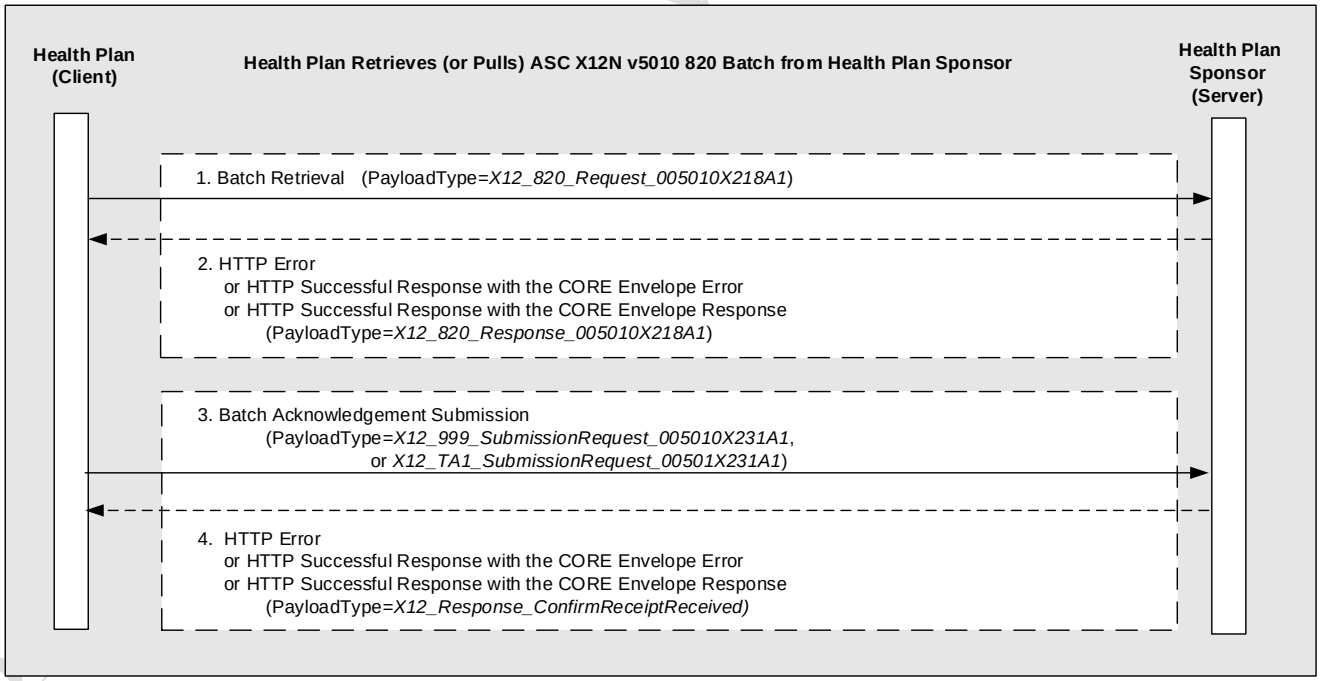


The following describes the *Benefit Enrollment and Maintenance* transaction using the *Generic Pull* interaction, as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Health Plan (Client) sends a Health Plan Sponsor (Server) a retrieval request for a Batch of Benefit Enrollment and Maintenance requests using PayloadType=*X12_834_Request_005010X220A1*.	Benefit Enrollment and Maintenance:
2	Health Plan Sponsor (Server) responds synchronously in Real time either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_834_Response_005010X220A1*)	Benefit Enrollment and Maintenance:

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
3	A Health Plan (Client) submits to a Health Plan Sponsor (Server) the acknowledgement (PayloadType *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
4	Health Plan Sponsor (Server) responds (synchronously to request message 3) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_Response_ConfirmReceiptReceived)*	Implementation Acknowledgement Submission

Example: Payroll Deducted and Other Group Premium Payment for Insurance Products (ASC X12N v5010 820)



1093

1094 The following describes the *Payroll Deducted and Other Group Premium Payment for Insurance Products*
1095 transaction using the *Generic Pull* interaction, as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Health Plan (Client) sends a Health Plan Sponsor (Server) a retrieval request for a Batch of *Payroll Deducted and Other Group Premium Payment for Insurance Products* using PayloadType=*X12_820_Request_005010X218A1*.	Payroll Deducted and Other Group Premium Payment for Insurance Products (Retrieval Response)
2	A Health Plan Sponsor (Server) responds synchronously in Real time either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_820_Response_005010X218A1*)	Payroll Deducted and Other Group Premium Payment for Insurance Products (Retrieval Response)
3	A Health Plan (Client) submits to a Health Plan Sponsor (Server) the acknowledgement (PayloadType=*X12_999_SubmissionRequest_005010X231A1* or *X12_TA1_SubmissionRequest_005010X231A1*) to a Health Plan. This acknowledgment submission is required by CORE Phase I and Phase II Rules.	Payroll Deducted and Other Group Premium Payment for Insurance Products (Batch Results Acknowledgment Submission)
4	A Health Plan Sponsor (Server) (responds synchronously to request message 3) to the request either with an: HTTP Error or HTTP Successful Response with the CORE Envelope Error or HTTP Successful Response with the CORE Envelope Response (PayloadType=*X12_Response_ConfirmReceiptReceived)*	Payroll Deducted and Other Group Premium Payment for Insurance Products (Batch Results Acknowledgment Response)

1096