CAQH CORE Connectivity Rule vC4.0.0
October 2024

Revision History for CAQH CORE Connectivity Rule vC4.0.0

Version	Revision	Description	Date
C4.0.0	Major	CAQH CORE Connectivity Rule vC4.0.0 balloted and approved via the CAQH CORE Voting Process.	December 2020
C4.0.0	Typographical	Technical formatting revisions to § 5.3.2.Specifications for REST API URI Path Endpoints for Payload Types (normative), §5.6 REST POST Message Structure (Informative Example) and §8.3 Sequence Diagrams for SOAP.	October 2024

Table of Contents

1. CAQH CORE Connectivity Rule: Background

1.1. CAQH CORE Overview

CAQH CORE is an industry-wide facilitator committed to the creation and adoption of healthcare operating rules that support standards, accelerate interoperability and align administrative and clinical activities among providers, health plans and patients. Guided by over 130 participating organizations – including providers, health plans, government entities, vendors, associations and standards development organizations – CAQH CORE Operating Rules drive a trusted, simple and sustainable healthcare information exchange that evolves and aligns with market needs.[1] CAQH CORE Operating Rules are developed using a consensus-based approach among industry stakeholders, and are designed to facilitate interoperability, improve utilization of administrative transactions, enhance efficiency and lower the cost of information exchange in healthcare. To date, this cross-industry commitment has resulted in operating rules that address many pain points of healthcare business transactions including: eligibility and benefits verification, claims and claims status, claim payment and remittance, health plan premium payment, enrollment and disenrollment, prior authorization and aspects of value-based healthcare such as patient attribution.

1.2. Evolution of CAQH CORE Connectivity

As stakeholders first began to implement HIPAA electronic transaction standards in the early 2000s, no operating rules existed to guide implementation. Health plans, healthcare providers and vendors defined key terms or the specific protocols for data sharing for themselves. The use of proprietary systems and workarounds had an effect opposite that intended by HIPAA administrative simplification provisions. As a result, administrative complexity and non-uniformity quickly rose and became the norm.

The industry solution was to establish CAQH CORE and task it with driving the creation and adoption of healthcare operating rules that support standards, accelerate interoperability and align administrative and clinical activities among providers, payers and consumers. In 2005, CAQH CORE began to develop CAQH CORE Connectivity Rules based on a consensus-driven process that brought diverse stakeholders together to establish protocols for implementing HIPAA and other standards, promoting interoperability across the industry.[2]

Initially, CAQH CORE Connectivity established base minimum requirements such as the use of Hypertext Transfer Protocol Secure (HTTPS) over the public internet and specified other aspects of connectivity and security including: acknowledgements, error handling and the CAQH CORE Connectivity “Safe Harbor”. Over the years, CAQH CORE Participating Organizations continued to develop and update CAQH CORE Connectivity requirements to enhance interoperability within the healthcare industry and five primary components of CAQH CORE Connectivity were ultimately established:

- **Safe Harbor:** The CAQH CORE Connectivity Safe Harbor specifies connectivity methods that application vendors, providers, and health plans can be assured will be supported by any HIPAA-covered entity. Since its inception, CAQH CORE Connectivity has used the public internet and HTTPS to facilitate information exchange, establishing a Safe Harbor connectivity method that ensures HIPAA-covered entities are capable and ready at the time of a request by a trading partner to exchange data using the CAQH CORE.
- **Transport:** CAQH CORE Connectivity addresses synchronous (Real Time Processing Mode) and asynchronous (Batch Processing Mode) message interaction patterns, which describe how connections are established and used for handling requests and responses.
- **Message Envelope:** CAQH CORE Connectivity includes a well-defined structure for organizing and formatting message envelope metadata, to identify the sender/receiver and ensure documents are delivered to the receiver.
- **Security**: CAQH CORE Connectivity requires communications using a secure and encrypted transport protocol to keep data transmission private.

---

[1] In 2012, CAQH CORE was designated by the Secretary of the Department of Health and Human Services (HHS) as the author for federally mandated operating rules under Section 1104 of the Patient Protection and Affordable Care Act (ACA).
[2] Per federal mandate, implementation of CAQH CORE Connectivity Rule vC1.1.0 and CAQH CORE Connectivity Rule vC2.2.0. is a requirement for all HIPAA-covered entities.

- **Authentication:** CAQH CORE Connectivity specifies authentication requirements to ensure correct identification of people, places and things as well as correct mapping between relevant resources and underlying systems.

These components reflect the types of communication elements needed to support interoperable exchanges of data and therefore are key elements of CAQH CORE Connectivity. Additionally, from a technical perspective, converging on common transport, envelope, security and authentication standards streamlines industry connectivity and reduces implementation variations to improve overall interoperability and efficiency of administrative transactions throughout the industry. Therefore, the established CAQH CORE Connectivity requirements enabled a framework for interoperability that is universal, easy to implement, low cost, secure, trusted, meaningful and industry recognized. Additional benefits of CAQH CORE Connectivity include:

- Specifying a "**Safe Harbor"** enabling baseline expectations for connectivity while **promoting flexibility** and supporting organizations at different levels of technology and maturity.
- Establishing a **secure and trusted method** for exchange of information over the Internet by providing security and authentication protocols.
- Enabling connectivity and its associated requirements to be **payload agnostic**, supporting a variety of data types and allowing for combability with existing and emerging standards.
- Supporting **error handling** using standard error codes to notify all parties whether a communication has occurred successfully.
- Promoting **ease of implementation** with connectivity schemas available at no cost.
- Enabling **direct lines of communication** with trading partners to minimize complexity and cost.
- Connectivity and security protocols and standards identified in the CAQH CORE Connectivity Rule for implementation are identified and voted on **for industry by industry**.
- **Widely implemented and accepted** across the industry.

However, as the healthcare industry progresses toward achieving alignment and interoperability across administrative and clinical systems, updating common methods of connectivity to include existing and emerging methods (e.g., Representational State Transfer REST Application Programming Interfaces (APIs), HL7 FHIR, etc.) for connectivity and data sharing could continue to ease administrative burden. Updated CAQH CORE Connectivity requirements offer an opportunity for operating rules to bridge the gap between existing and emerging standards and achieve alignment to support administrative and clinical data exchange, setting the course for long-term industry operability.

2. Issues to be Addressed and Business Justification

While prior versions of CAQH CORE Connectivity established a national base guiding healthcare communication of administrative data, the connectivity environment of today adds additional levels of operational complexity and elevated costs for stakeholders who have implemented a multitude of connectivity methods. These methods are based on open standards and proprietary approaches, to facilitate the exchange of administrative and clinical healthcare data. Over the years, this created a fragmented connectivity ecosystem where senders and receivers of electronic data are required to support multiple communications channels and protocols, adding additional levels of operational complexity and elevated cost that reduce interoperability.

The diversity of connectivity methods used to exchange healthcare data ranges from high-speed, dedicated lines to low-speed dial-up lines into bulletin board/web portal-type systems, as well as File Transfer Protocol (FTP), Virtual Private Network (VPN), HTTPS and Web Services over the Internet. Each of these connectivity modes can be either direct between trading partners or to intermediaries such as clearinghouses, that serve as switches/hubs or provide other services for both providers and plans.

Recognizing that the healthcare industry uses multiple connectivity methods for electronic administrative transactions, prior versions of the CAQH CORE Connectivity requirements aimed to address the gap by formulating connectivity and security requirements to support healthcare industry specific transactions. CAQH CORE Connectivity has been successful in promoting interoperability for the exchange of administrative data. However, the industry continues to require alignment on a common set of communication protocols. Expanding the existing CAQH CORE Connectivity requirements will bridge connectivity and interoperability barriers between administrative and clinical systems and incorporate both new and emerging standards as technology continues to evolve within the industry.

2.1. Update to the CAQH CORE Connectivity Rules

Despite progress made by the healthcare industry over the years to resolve interoperability challenges, the presence of a fragmented foundational connectivity ecosystem for the intersection of administrative and clinical data continues to inhibit successful, industry-wide interoperability. As such, industry support and commitment for common, uniform and consistent connectivity approaches has strengthened – a major goal of the 21st Century Cures Act is to achieve nationwide interoperability. To help realize this goal, Office of the National Coordinator for Health Information Technology (ONC) and Centers for Medicare & Medicaid Services (CMS) published rules to facilitate patient access to information through APIs including administrative data that has historically been shared between health plans and providers through different methods. Common, uniform and consistent connectivity approaches, including the use of APIs, could support a broader range of use cases connecting clinical and administrative data across stakeholder groups including patients, providers, health plans, and vendors to achieve industry-wide interoperability.

CAQH CORE has led efforts to move the industry forward in approaches towards interoperability via CAQH CORE Connectivity. As an example, in prior connectivity rule development efforts CAQH CORE evaluated opportunities for exploring support for Representational State Transfer (REST) APIs with CAQH CORE Participating Organizations.

In response to current industry interest and need, CAQH CORE Participating Organizations, representing a diverse mix of provider, health plan, vendor and government entities convened to consider updates to the CAQH CORE Connectivity requirements. The goal of the update to the CAQH CORE Connectivity requirements was to move the industry towards a common set of Safe Harbor connectivity methods that address existing and emerging standards and protocols to support alignment needed for administrative and clinical data exchange.

From February 2020 to August 2020, CAQH CORE Participating Organizations completed feedback forms, straw polls and ballots and participated in discussions to agree upon key enhancements to the CAQH CORE Connectivity requirements. Ultimately, six opportunity areas for enhancement of the CAQH CORE Connectivity Rule requirements were identified and pursued:

#	Opportunity Area	CAQH CORE Connectivity Rule Enhancements
1	Uniform CAQH CORE Connectivity Rule	▪ Publication of a single CAQH CORE Connectivity Rule that addresses all published CAQH CORE Operating Rules.
2	CAQH CORE Safe Harbor Requirements	▪ Establishment of Safe Harbor provisions that HIPAA covered entities must have the capability to support requirements addressed by CAQH CORE Connectivity Rule vC.4.0.0
3	CAQH CORE Connectivity Transport Security Requirements	▪ Sunset requirements that specify the use of Secure Socket Layer (SSL). ▪ Update requirements that specify the use of Transport Layer Security (TLS), requiring the use of TLS 1.2 or higher.
4	CAQH CORE Authorization Requirements	▪ Addition of requirements to support authorization by requiring the use OAuth 2.0.
5	CAQH CORE Message Interaction Requirements	▪ Inclusion of message interaction patterns for attachment transactions.
6	CAQH CORE Connectivity Web Service Requirements	▪ Continuation of support for the exchange of SOAP messages. ▪ Addition of a rule requirements to support the exchange of REST messages. ▪ Requirement to support JSON as a REST API Interface format.

#	Opportunity Area	CAQH CORE Connectivity Rule Enhancements
		▪ Requirements to support a specific set of HTTP Methods to indicate a desired action to be performed for a given resource. ▪ Requirement to support versioning management of REST APIs and the CAQH CORE Connectivity Rule. ▪ Specification of normative naming conventions for API Endpoints. ▪ Requirements to support a base set of metadata required to be used for the exchange of REST messages.

These updates to the CAQH CORE Connectivity Rule serve as a bridge between the existing and emerging standards and protocols to ensure industry interoperability needs are met. Furthermore, the update represents a key step in modernizing the national floor for guiding connectivity expectations throughout the industry. However, as with all CAQH CORE Operating Rules, the requirements in this CAQH CORE Connectivity Rule are a floor and not a ceiling in terms of what organizations can implement.

3. Scope

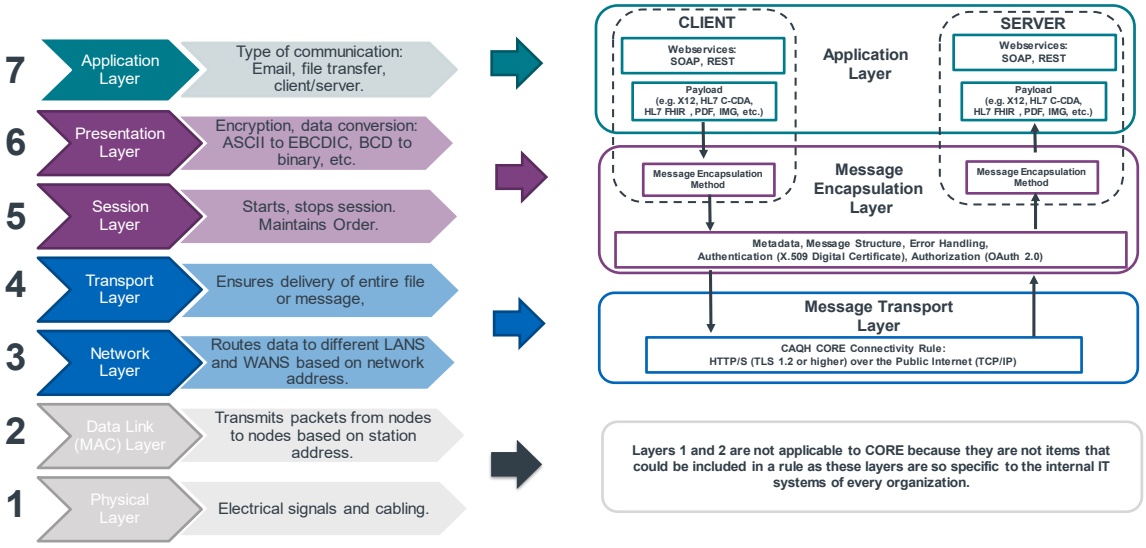
3.1. What the Rule Applies To

The technical scope of this CAQH CORE Connectivity Rule can be described in terms of the specific network layers within the Open Systems Interconnection Basic Reference Model[3] (OSI model). As shown in the diagram below, the scope of this CAQH CORE Connectivity Rule is OSI Layers 3 and 4 (Network and Transport) and OSI Layers 5 and 6 (Session and Presentation layers, also called Message Encapsulation layers). As shown in the Figure 3.1.1, typically an application file (or Payload) such as X12 or HL7 is created or processed by an application that resides in the Application Layer (Layer 7 in the OSI Model). The Message Encapsulation layer (Layers 5 and 6 in the OSI Model) creates 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).

The CAQH CORE Connectivity Rule vC1.1.0 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: X12, HL7 clinical messages, zipped files, etc.

[3] 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



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[4]
- TLS 1.2 or higher.
    - This does not preclude the optional use of TLS 1.3 (or a higher version) for connectivity with trading partners whose security policies require the enhanced security afforded by TLS 1.3 or higher.
- SOAP Version 1.2 or higher
- WSDL Version 1.1 or higher
- JavaScript Object Notation (JSON)[5]
- X.509 Digital Certificate addressing authentication[6]
- OAuth 2.0 or higher addressing authorization[7]

[4] Hereafter the combination of HTTP and TLS is referenced as HTTP/S.
[5] JavaScript Object Notation (JSON), is a minimal, readable format for structuring data. It is used primarily to transmit data between a server and web application.
[6] X.509 is defined by the International Telecommunications Union's Standardization sector (ITU-T), and is based on ASN.1, another ITU-T standard
[7] OAuth 2.0 is the industry-standard protocol for authorization https://tools.ietf.org/html/rfc6749

3.3. When the Rule Applies

The CAQH CORE Connectivity Rule vC4.0.0 applies when trading partners are exchanging any of the following X12 transactions specified in CAQH CORE Operating Rules[8]:

- X12 v5010 270/271 Healthcare Eligibility Benefit Inquiry and Response[9]
- X12 v6020X314 275 Additional Information to Support a Health Care Claim or Encounter
- X12 v6020X316 275 Additional Information to Support a Health Care Services Review
- X12 v5010X212 276/277 Health Care Claim Status Request and Response[10]
- X12 v5010X213 277 Health Care Claim Request for Additional Information
- X12 v6020X313 277 Health Care Claim Request for Additional Information
- X12 v5010X214 277 Health Care Claim Acknowledgement
- X12 v5010X215 278 Health Care Services Review - Inquiry and Response
- X12 v5010X216 278 Health Care Services Review - Notification and Acknowledgment
- X12 v5010X217 278 Health Care Services Review - Request for Review and Response
- X12 v5010X218 820 Payroll Deducted and Other Group Premium Payment for Insurance Products
- X12 v5010X220 834 Benefit Enrollment and Maintenance
- X12 v5010X307 834 Health Insurance Exchange Enrollment
- X12 v5010X318 834 Plan Member Reporting
- X12 v5010X221 pic Health Care Claim Payment/Advice[11]
- X12 v5010X223 837I Health Care Claim – Institutional
- X12 v5010X222 837P Health Care Claim – Professional
- X12 v5010X224 837D Health Care Claim – Dental
- X12 v5010X231 999 Implementation Acknowledgement for Health Care Insurance
- X12 TA1 Interchange Acknowledgement

This rule may also be applied to other payload types (e.g., HL7 C-CDA, .pdf, .doc, etc.). Note: some entities may also apply this rule to other X12 administrative transactions. This 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. HIPAA-covered entities and their agents may also use this rule for the exchange of eligibility, claim status and ERA transactions in accordance with the Safe Harbor provision of the CAQH CORE Connectivity Rule vC2.2.0, which is ACA-mandated. However, this does not permit any HIPAA-covered entity and its agent to discontinue support for the exchange of transactions addressed in previous versions of CORE Connectivity as required in the CAQH CORE Connectivity Rule vC2.2.0.

3.4. When the Rule Does Not Apply

This rule is designed to be payload agnostic, meaning that the SOAP and REST services are not aware of the content they are serving, and as such it is expected that HIPAA-covered entities and their agents will use this methodology for other payloads as described in §3.3; however, the rule does not require this.

---

[8] To create a uniform CAQH CORE Connectivity Rule that applies across all CORE Operating Rules, this section includes all X12 transactions addressed by voluntary and mandated CAQH CORE Operating Rules and Operating Rules that are in development.

[9] HIPAA-covered entities and their agents may also use this CAQH CORE Connectivity Rule for the exchange of HIPAA-mandated eligibility, claim status and electronic remittance advice transactions in accordance with the Safe Harbor provision of the ACA-mandated CAQH CORE Connectivity Rule vC2.2.0. However, this does not permit any HIPAA-covered entity or its agent to discontinue support for the exchange of transactions addressed in CAQH CORE Operating Rules as required in the ACA-mandated CAQH CORE Connectivity Rule vC2.2.0.

[10] Ibid.

[11] Ibid.

3.5. What the Rule Does Not Require

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

Further, this 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.
- Additional centralized OAuth services to an OAuth client for "secure delegated access" to server resources.
- 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).
- OAuth Authorization is a local decision at the site that receives a request.
- The list of trusted X.509 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 and §5.7)

3.7. CAQH CORE-required Processing Mode and Payload Type Tables

This 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 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. CAQH CORE-required Processing Mode Table

The 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. CAQH CORE-required Payload Type Table

The 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 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.

3.8. Rule Maintenance

3.8.1. Maintenance of Connectivity Standards Used in this Rule

The connectivity standards used in this rule (See §3.2) were determined by CAQH CORE Participating Organizations to be the most appropriate to implement at the time this rule was approved. CAQH CORE recognizes that as technology continues to evolve and mature, the connectivity standards used in this rule may require modification and updates to meet emerging or new industry needs. Given this anticipated need for maintenance activity to the connectivity standards used in this rule, CAQH CORE understands that a process and policy to enable the review and maintenance of the connectivity standards used in this rule on a regular basis is required[12].

Such review and maintenance of the connectivity standards used in this rule will follow standard CAQH CORE processes for rule revisions.[13] CAQH CORE will develop such a process and policy for the first review of the connectivity standards used in this rule in accordance with CAQH CORE Guiding Principles following the approval of this rule. The first review may commence:

- As determined by CAQH CORE processes for rule revisions

  Or

- One year after the passage of a Federal regulation requiring implementation of this 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 connectivity standards used in this rule will be reviewed and approved by CAQH CORE as necessary to ensure accurate and timely revision.

3.8.2. Maintenance of the CAQH 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 CAQH CORE-required Processing Mode Table or the 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, *COREProcessingModePayloadTypeTablesC4.0.0.docx,* will be developed by CAQH CORE.

[12] CAQH CORE Change Process and Maintenance
[13] CAQH CORE has restructured its operating rules from phase-based rule sets to rule sets based on business transactions which creates a flexible framework for adding new rules/requirements, updating existing operating rules, and removing outdated requirements for each business transaction.

Such review and maintenance of either the CAQH CORE-required Processing Mode Table or the CAQH CORE-required Payload Type Table or both will follow standard CAQH CORE processes for rule revisions.[14] 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 this rule. The first review may commence:

- As determined by CAQH CORE processes for rule revisions

  Or

- One year after the passage of a Federal regulation requiring implementation of this 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 CAQH CORE Infrastructure Rules will be considered during the review of potential revisions. 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.

3.9. Assumptions

The following assumptions apply to this rule:

- Interoperability, utilization, and efficiency will improve by having fewer connectivity/security variations, uniform enveloping standards and metadata, and uniform REST API specifications.
- 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 designed to apply across all CAQH CORE Operating Rules.
- The CAQH CORE Guiding Principles apply to this rule and all other rules.

4. SOAP Rule

This section specifies the requirements for transport, message envelope, authentication, authorization, envelope metadata and the specifications for SOAP+WSDL.

4.1. CAQH CORE Authentication, Authorization, and Message Envelope Requirements

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

4.1.1. Authentication Requirement

This rule requires HIPAA-covered entities and their agents, including health plans and healthcare providers and their respective agents, to support the use of X.509 Mutual Authentication over TLS 1.2 or higher.

4.1.2. Authorization Requirement

This rule requires HIPAA-covered health plans and their agents to support OAuth 2.0 Client Authorization over TLS 1.2 or higher.

HIPAA-covered providers and their agents may optionally use OAuth 2.0 Client Authorization over TLS 1.2 or higher.

[14] Ibid.

4.1.3. Message Envelope Requirement

This rule requires the use of SOAP+WSDL.

4.1.3.1. Specifications for SOAP+WSDL Envelope Standard (normative[15])

This section defines the SOAP+WSDL envelope method for this rule. 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*.

4.1.3.2. CAQH CORE Connectivity Rule vC4.0.0 XML Schema Specification (normative)

The CAQH CORE compliant XML Schema Specification file name below is called *CORERuleC4.0.0.xsd*, and is available at *http://www.caqh.org/sites/default/files/core/wsdl/CORERuleC4.0.0.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*)
- 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 CAQH CORE Safe Harbor provision.

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

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.caqh.org/SOAP/WSDL/CORERuleC4.0.0.xsd"
targetNamespace="http://www.caqh.org/SOAP/WSDL/CORERuleC4.0.0.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>
```

4.1.3.3. CAQH CORE Connectivity Web Services Definition Language (WSDL) Specification (normative)

The CAQH CORE Connectivity SOAP Rule vC4.0.0 Web Services Definition Language (WSDL) file below is called *CORERuleC4.0.0.wsdl*, and is available at *http://www.caqh.org/sites/default/files/core/wsdl/CORERuleC4.0.0.wsdl*. The WSDL below makes use of the XML Schema (*CORERuleC4.0.0.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 SOAP Rule vC4.0.0 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/CORERuleC4.0.0.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/CORERuleC4.0.0.xsd"
schemaLocation="CORERuleC4.0.0.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[16]. They are based on the real-world examples provided by CAQH CORE Participating Organizations but have been updated to use the CAQH CORE-required metadata that is part of CAQH CORE Connectivity Rule vC4.0.0.

4.1.3.4. 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 CAQH CORE Connectivity Rule vC4.0.0. (See §4.4)
3. The Real Time Payload file (MTOM attachment) is shown colored in orange.

[16] A non-normative description is informational only. See §7.2 Abbreviations and Definitions Used in this Rule.

```
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/CORERuleC4.0.0.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>C4.0.0</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--
```

4.1.3.5. 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 CAQH CORE Connectivity SOAP Rule vC4.0.0. (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/CORERuleC4.0.0.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>C4.0.0</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--
```

4.1.3.6. Batch Submission Message (non-normative)[17]

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 CAQH CORE Connectivity SOAP Rule vC4.0.0. (See §4.4)
3. The Batch file (MTOM attachment) is shown colored in orange.

[17] 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/CORERuleC4.0.0.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>C4.0.0</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--
```

4.1.3.7. Batch Submission Response Message (non-normative)[18]

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 CAQH CORE Connectivity SOAP Rule vC4.0.0. (See §4.4)

[18] 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.

```
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/CORERuleC4.0.0.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>C4.0.0</CORERuleVersion>
                        <ErrorCode>Success</ErrorCode>
                        <ErrorMessage></ErrorMessage>
                 </ns1:COREEnvelopeBatchSubmissionResponse>
        </soapenv:Body>
 </soapenv:Envelope>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339--
```

4.1.3.8. Batch Submission Acknowledgement Retrieval Request Message (non-normative)

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

1. The HTTP Headers are shown colored in blue.
2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CAQH CORE Connectivity SOAP Rule vC4.0.0. (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="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/CORERuleC4.0.0.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>C4.0.0</CORERuleVersion>
                </ns1:COREEnvelopeBatchSubmissionAckRetrievalRequest>
        </soapenv:Body>
</soapenv:Envelope>
--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```

4.1.3.9. Batch Submission Acknowledgement Retrieval Response Message (non-normative)[19]

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

1. The HTTP Headers are shown colored in blue.
2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CAQH CORE Connectivity SOAP Rule vC4.0.0. (See §4.4)
3. The MTOM Attachment is colored in orange.

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

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

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Body>
                <ns1:COREEnvelopeBatchSubmissionAckRetrievalResponse
                xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERuleC4.0.0.xsd">
                        <PayloadType>X12_999_Response_005010X231A1</PayloadType>
```

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

```
                    <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: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--
```

4.1.3.10. Batch Results Retrieval Request Message (non-normative)[20]

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 CAQH CORE Connectivity SOAP Rule vC4.0.0. (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/CORERuleC4.0.0.xsd">
    <PayloadType>X12_278_Request_Batch_Results_005010X217E1_2</PayloadType>
```

[20] 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.

```
                    <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--
```

4.1.3.11. Batch Results Retrieval Response Message (non-normative)[21]

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

1. The HTTP Headers are shown colored in blue.
2. The portion of the SOAP envelope colored in green has the metadata that is defined as part of the CAQH CORE Connectivity SOAP Rule vC4.0.0. (See §4.4)
3. The MTOM Attachment is colored in orange.

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

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

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Body>
                <ns1:COREEnvelopeBatchResultsRetrievalResponse
                xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERuleC4.0.0.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>C4.0.0</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>
```

[21] 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.

```
                </ns1:COREEnvelopeBatchResultsRetrievalResponse>
        </soapenv:Body>
 </soapenv:Envelope>

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

<Response batch file>
--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339--
```

4.1.3.12. Batch Results Acknowledgement Submission Message (non-normative)[22]

The Batch Results Acknowledgement Submission message structure shown below specifies SOAP 1.2, and also uses MTOM (See §7.2) 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 CAQH CORE Connectivity SOAP Rule vC4.0.0. (See §4.4)
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/CORERuleC4.0.0.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>C4.0.0</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>
```

[22] 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.

```
 </soapenv:Envelope>

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

<999 file>

--MIMEBoundaryurn_uuid_5117AAE1116EA8B87A1200060184614--
```

4.1.3.13. Batch Results Acknowledgement Submission Response Message (non-normative)[23]

The Batch Results Acknowledgement 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 CAQH CORE Connectivity SOAP Rule vC4.0.0. (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/CORERuleC4.0.0.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>C4.0.0</CORERuleVersion>
                         <ErrorCode>Success</ErrorCode>
                         <ErrorMessage></ErrorMessage>
                 </ns1:COREEnvelopeBatchResultsAckSubmissionResponse>
        </soapenv:Body>
 </soapenv:Envelope>

--MIMEBoundaryurn_uuid_0B72121B1FEFA9BDD31200060195339--
```

4.1.3.14. Error Message Structure (non-normative)

[23] 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.

The Error message structure shown below uses the SOAP Fault specifications within SOAP 1.2. As described in §4.2.4, SOAP Faults must be used to send errors at the SOAP level. The HTTP Headers are shown colored in blue. The remainder of the request is the SOAP Envelope.

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


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

<soapenv:Header>

</soapenv:Header>

<soapenv:Body>

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

</soapenv:Envelope>
```

4.1.3.15. Envelope Processing Error Message (non-normative)

The Error message structure shown below illustrates a SOAP-based message that indicates an error has occurred within processing the envelope. The HTTP Headers are shown colored in blue. The remainder of the request is the SOAP Envelope. The envelope structure and metadata that is defined within CAQH CORE Connectivity SOAP Rule vC4.0.0 is colored in green.

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

<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope">
        <soapenv:Body>
                <ns1:COREEnvelopeRealTimeResponse
                xmlns:ns1="http://www.caqh.org/SOAP/WSDL/CORERuleC4.0.0.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>C4.0.0</CORERuleVersion>
                        <Payload></Payload>
        <ErrorCode>VersionMismatch</ErrorCode>
                        <ErrorMessage>Expecting Version X, received Version Y</ErrorMessage>
                </ns1:COREEnvelopeRealTimeResponse>
        </soapenv:Body>
 </soapenv:Envelope>
```

4.1.4. Real Time and Batch Payload Attachment Handling

Payload must be sent as an MTOM[24] encapsulated object.

4.2. General Specifications Applicable to the SOAP Envelope Method

4.2.1. Required Transport Method

HIPAA-covered entities and their agents must be able to implement HTTP/S Version 1.1 over the public Internet as a transport method. 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 X12 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 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 X12 message response (e.g., X12 005010X231A1 Implementation Acknowledgement for Health Care Insurance (999) [hereafter X12 v5010 999]

Or

- The corresponding X12 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 X12 response in the timeframes specified in the transaction's corresponding 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 X12 response, such as an X12 v5010 999 in the HTTP response to the batch request, even if their systems' capabilities allow such a response. All X12 responses must be available for pick up by the message sender (client) in accordance with the respective CAQH CORE Infrastructure Rule for the transaction.

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 X12 response in the timeframes specified in the transaction's corresponding CAQH CORE Infrastructure Rule.

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

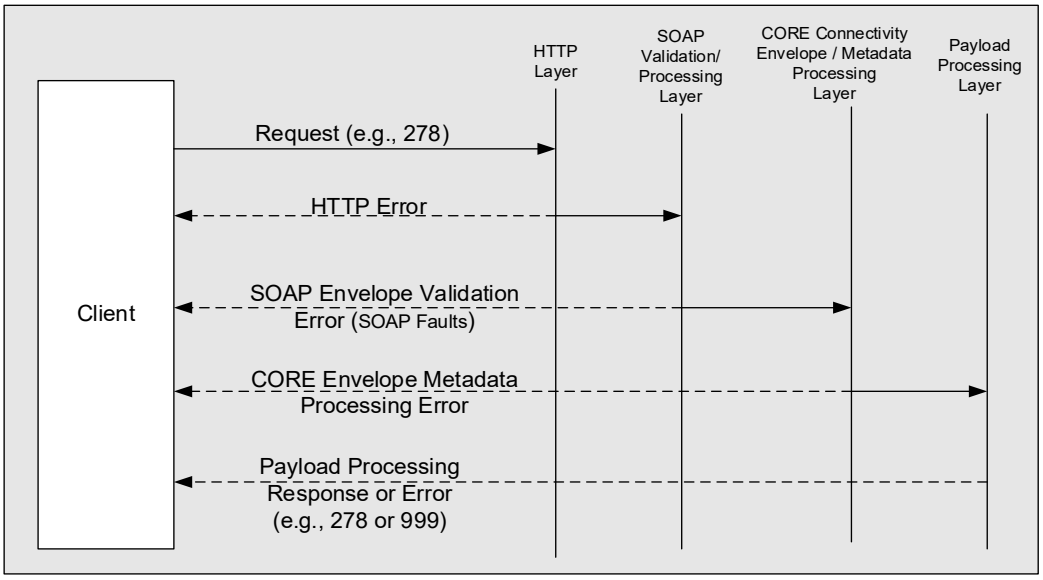
4.2.6. Error Handling

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 webserver)
- Validating the SOAP Envelope (can be handled by messaging middleware or integration brokers)
- Processing the CORE specific metadata located in the SOAP Envelope
- Processing the Payload (e.g., X12, typically handled by application business logic)

Once a request (e.g., X12 v5010 278 Request) is submitted it goes through these four 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 client), which may be returned to the client. 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., X12 v5010 278 Response or X12 v5010 999 or X12 TA1).

Figure #4.2.6



Note: In Figure #4.2.6 above, the dotted line arrows indicate error messages being returned to the client 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[25])

The processing and error codes for the HTTP Layer are defined as part of the HTTP specifications [*http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html*]. The intended use of these status and error codes in

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

processing the requests are specified in Table 4.2.6.1 and are consistent with the HTTP status codes from CAQH CORE Connectivity Rule vC1.1.0 (formerly 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 CAQH CORE Connectivity error handling.

Table 4.2.6.1	
HTTP Status/Error Codes (Normative, Not Comprehensive)	**CAQH CORE Rule Specific Description (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 webserver encountered a processing error
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 CAQH 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 CAQH CORE Connectivity error handling.

Table 4.2.6.2	
SOAP Faults (Normative; Not Comprehensive)	**CAQH 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.

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

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

Table 4.2.6.3	
CORE-compliant Envelope Processing Status/Error Codes (Normative, Comprehensive)	**CAQH CORE Status Code Description[26] (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 CAQH 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 Connectivity.
VersionMismatch	The CAQH 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)
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 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 CAQH 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

[26] 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].

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 CAQH 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 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 CAQH 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 and 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 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 CAQH 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 and their agents must attest that their capacity planning addresses the above three factors that affect large volume single Real Time processing[27]. HIPAA-covered entities and 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.

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

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 CAQH CORE response time rule's performance criteria, and/or other appropriate action.

4.2.9.2. Batch Transactions

The HIPAA-covered entity and 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 CAQH CORE Rule.

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

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

HIPAA-covered entities and their agents must attest that their capacity planning addresses the above three 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 CAQH CORE Infrastructure Rule requirements.

- If a Real Time response message is not received within the 60 second response period, the 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 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 client system must notify the client to contact the server 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 web site. CAQH 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 CAQH CORE rules.

- CAQH 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 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 policies using X.509 Client Certificates (e.g., how to enroll and obtain a Client Certificate to connect to that receiver (server).
- Authorization policies using OAuth 2.0 Tokens (e.g., how to enroll and obtain a Token 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 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 TLS 1.2 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 Message Envelope SOAP+WSDL. With the exception of *ErrorCode* and *ErrorMessage* fields, which are only sent in the response, the CAQH CORE required envelope metadata for the request and response are required to be identical.

4.4.1. Message Envelope

The CAQH CORE Connectivity Rule v1.1.0 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 (X12), HL7 clinical messages, zipped files, etc.

4.4.2. Table of CAQH CORE Envelope Metadata

Table 4.4.2 CAQH CORE Envelope Metadata						
Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[28]**	**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., X12N transaction set 837, 820, 278, etc.).	• Message routing • Efficient processing • Auditing	PayloadType	Required for both	Coded Set	Please see CAQH CORE-required Payload Type Table document for enumeration of PayloadType field.
Processing Mode	Processing Mode indicates Batch or Real Time[29] 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 [30] Shall not be used for Real time.	Integer (Base 10)	

[28] 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.
[29] See *Appendix 7.2: Abbreviations and Definitions used in this Rule* for a definition of Batch and Real Time.
[30] 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.

Table 4.4.2 CAQH CORE Envelope Metadata						
Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[28]**	**Requirement Indicator for Real Time and Batch**	**DataType**	**Field Constraints or Value-sets (Not Comprehensive)**
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[31], to ensure uniqueness.
Time Stamp	The Sender (request) or Receiver (response) Time Stamp. This does not require a shared time server for consistent time.	• Auditing • Troubleshooting	TimeStamp	Required for both	dateTime	dateTime (http://www.w3.org/TR/xmlschema11-2/#dateTime)

[31] 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.

Table 4.4.2 CAQH CORE Envelope Metadata						
Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[28]**	**Requirement Indicator for Real Time and Batch**	**DataType**	**Field Constraints or Value-sets (Not Comprehensive)**
Sender Identifier	A unique[32] business entity identifier representing the message envelope creator. Sender Identifier is better suited for identifying business entities and trading partners than Username because: • Username is usually anonymized for security reasons and to protect privacy. • Username attribute does not exist if another authentication method is used. • Authentication and messaging may happen on different layers[33] 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.

[32] Unique within the Sender's (client's) domain.

Table 4.4.2 CAQH CORE Envelope Metadata						
Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[28]**	**Requirement Indicator for Real Time and Batch**	**DataType**	**Field Constraints or Value-sets (Not Comprehensive)**
Receiver Identifier	A unique[34] 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	C4.0.0
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 [35] Not used for Real Time	String	Algorithm is SHA-1[36] 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.

[34] Unique within a Receiver's (server's) domain.
[35] 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.
[36] 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.

Table 4.4.2 CAQH CORE Envelope Metadata						
Non-normative (Descriptive)			Normative (Definitive)			
Element	**Description**	**Expected Use**	**Field Name[28]**	**Requirement Indicator for Real Time and Batch**	**DataType**	**Field Constraints or Value-sets (Not Comprehensive)**
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 and its agent must support the transaction processing mode requirements (i.e., Real Time and/or Batch) as specified in the *COREProcessingModePayloadTypeTablesC4.0.0.docx* companion document to this CAQH CORE Connectivity SOAP Rule vC4.0.0 when exchanging transactions in conformance with this CAQH CORE Connectivity SOAP Rule vC4.0.0.

The Processing Mode requirements specified in the CAQH CORE-required Processing Mode Table also apply when a HIPAA-covered entity and 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 X12 Payloads (Normative)

A HIPAA-covered entity and 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 *COREProcessingModePayloadTypeTablesC4.0.0.docx* companion document to CAQH CORE Connectivity SOAP Rule vC4.0.0. (See Table 4.4.2, and §6). (See §3.8.2 for maintenance requirements.)

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

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

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

Note: SDO stands for Standards Development Organization.

For example, an HL7 V2 based ADT04 Version 2.3.1 payload may specify the *PayloadType* as *HL7_ADT04_2_3_1*.

5. REST Rule

This section specifies the requirements for transport, web services Application Programming Interface (API), authentication and authorization specifications for a RESTful interface performed directly on the server resource using a HTTP request/response.

5.1. CAQH CORE REST API Interface Format, Authentication and Authorization Requirements

This rule requires HIPAA-covered entities and their agents to support only one set of requirements for a REST API Interface format, one method for authorization, and one method for authentication in order to reduce variations and enable greater interoperability in the market.

5.1.1. REST API Interface Format Requirement

This rule requires the use of JavaScript Object Notation (JSON) for REST Interfaces.

5.1.2. Authentication Requirement

This rule requires HIPAA-covered entities, including health plans and healthcare providers and their respective agents, to support the use of X.509 Mutual Authentication over TLS 1.2 or higher.[37]

5.1.3. Authorization Requirement

This rule requires HIPAA-covered entities and their agents to support the use of OAuth 2.0 (mutual certificate-based authorization) over TLS 1.2 or higher.[38]

5.2. General Specifications Applicable to REST APIs

5.2.1. Required Transport Method

HIPAA-covered entities and their agents must be able to implement HTTP/S Version 1.1 over the public Internet as a transport method. 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.

5.2.2. 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 X12 messages in both Batch (asynchronous) and Real Time (synchronous) Processing Modes, but the response message from the receiver (server) is different depending on whether the sender's (client's) message is a Synchronous Real Time request, Asynchronous Batch submission, or Asynchronous Batch request pickup. This rule supports both Synchronous Real-time and Asynchronous Batch Processing for the transport of REST exchanges.

5.2.3. Synchronous Real Time Requests

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

- A transport or error response (See §4.2.6)

  Or

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

[37] The Internet Engineering Task Force describes OAuth client authentication and certificate-bound access and refresh tokens using mutual Transport Layer Security (TLS) authentication with X.509 certificates: https://tools.ietf.org/html/rfc8705.
[38] Ibid.

Or

- The corresponding X12 v5010 response transaction to the submitted request.

5.2.4. Asynchronous Batch Submission

In this Processing Mode the response differs because message receivers (servers) are not required to provide a corresponding X12 response in the timeframes specified in the transaction's corresponding CAQH CORE Infrastructure Rule for Real Time.

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

5.2.5. Asynchronous Batch Response Pick Up

Asynchronous Batch responses must be picked up after the message receiver (server) has processed an Asynchronous Batch submission corresponding to the X12 response in the timeframes specified in the transaction's corresponding CAQH CORE Infrastructure Rule.

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

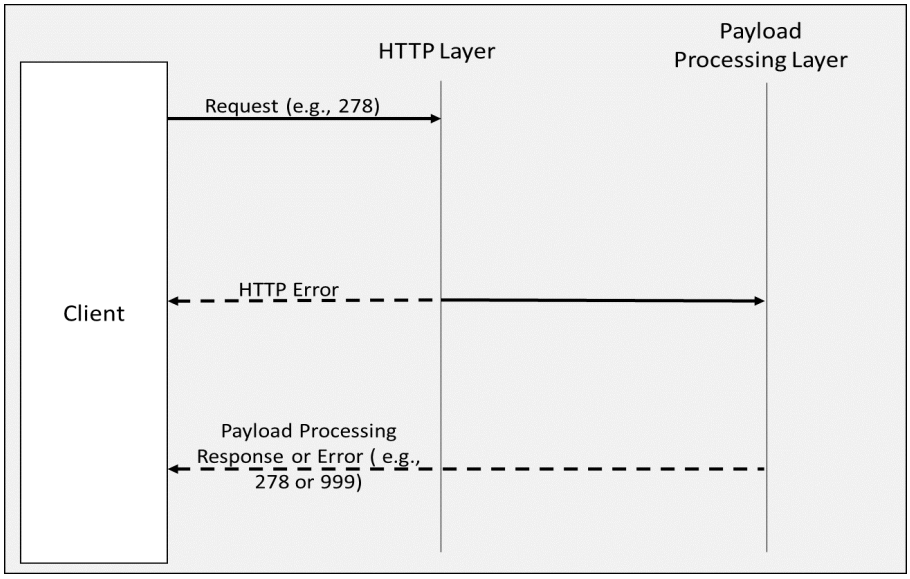
5.2.6. Error Handling

When the message sender (client) makes a request to the message receiver (server) using HTTP/S and the message receiver successfully receives the request, the message receiver must notify the message sender if the request was successfully handled at each of the of the following stages:

- Processing of HTTP headers (typically handled by a webserver)
- Processing the Payload (e.g., X12, typically handled by application business logic)

Once a request (e.g., X12 v5010 278 Request) is submitted it goes through these layers identified in Figure #5.3.6 below. At each of these layers, some part of the request is processed. There can be errors at either layer (indicated by the dotted arrows being returned to the client), which may be returned to the client. If there is an error in processing the message at either layer, the request does not get passed on. If no errors are encountered at a layer, the request is passed to the next processing layer. Once the Payload Processing Layer processes the payload, it returns a response or error (e.g., X12 v5010 278 Response, X12 v5010 999 or X12 TA1).

Figure #5.2.6



Note: In Figure #5.2.6 above, the dotted line arrows indicate error messages being returned to the Client if there is a processing error at the corresponding logical processing layer. The straight-line arrows indicate the request and response messages.

5.2.6.1. HTTP Status and Error Codes (Normative, Not Comprehensive[39])

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 previous version of the CAQH CORE Connectivity Rule.

The status and error codes included in Table 5.2.6.1 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 CAQH CORE Connectivity error handling.

[39] 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 5.2.6.1	
HTTP Status/Error Codes (Normative, Not Comprehensive)	**CAQH CORE Rule Specific Description [40] (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
401 Unauthorized	Client failed to authenticate with the server
403 Forbidden	Access denied
404 Not Found	The requested resource does not exist
429 Rate Limit Exceeded	The user has sent too many requests in a given amount of time
500 Internal Server Error	The webserver encountered a processing error
5xx Server errors	Standard set of server-side errors (e.g., 503 Service Unavailable)

5.2.7. Audit Handling

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

5.2.8. Tracking of Date and Time and Payload

In order to comply with the corresponding transaction's CAQH CORE Infrastructure Rules message receivers (servers) are required to track the times of any received inbound messages and respond with the outbound message for that Payload. In addition, as specified in the CAQH CORE REST HTTP Metadata Table 5.5, message senders (clients) must include the date and time the message was sent and last modified. While other elements may also be used for reporting and auditing purposes, date, time, and payload are minimally required elements.

5.2.9. Capacity Plan

5.2.9.1. Synchronous Real Time Transactions

A HIPAA-covered entity and its agent must have a capacity plan such that it can receive and process a large number of single concurrent Synchronous 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 CAQH CORE Infrastructure Rule.

Three major factors affect the specific number of Large Volume of Single Synchronous Real Time Transactions capable of being transported and processed within a given CAQH CORE response time frame. They are:

1. The amount of message metadata and message 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 and their agents must attest that their capacity planning addresses the above three factors that affect large volume single Synchronous Real Time processing. HIPAA-covered entities and 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.

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

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 CAQH CORE response time rule's performance criteria, and/or other appropriate action.

5.2.10. Synchronous Real Time Response, Timeout and Retransmission Requirements

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

- If a Synchronous Real Time response message is not received within the 60 second response period, the client system should send a duplicate transaction no sooner than 90 seconds after the original attempt was sent.
- If no Synchronous Real Time response is received after the second attempt, the 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 client system must notify the client to contact the server directly to determine if system availability problems exist or if there are known Internet traffic constraints causing the delay.

5.2.11. Asynchronous Batch Transactions

The HIPAA-covered entity and its agent's messaging system must have the capability to receive and process large Batch transaction files if the entity supports Asynchronous 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 CAQH CORE Infrastructure Rule.

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

4. 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;
5. The characteristics of the message handling software and its conciseness of design and coding; and,
6. The architecture of the intervening hardware, software and communication platform.

HIPAA-covered entities and their agents must attest that their capacity planning addresses the above three factors that affect large Asynchronous Batch processing. The maximum number of transaction sets to be included in a large Asynchronous Batch file is determined between trading partners.

5.3. Specifications for REST API Uniform Resource Identifiers (URI) Paths

5.3.1. Specifications for REST API URI Path Versioning (normative)

This rule requires message receivers (servers) to communicate the version of the CAQH CORE Connectivity Rule implemented and version of the REST API through the URI Path.

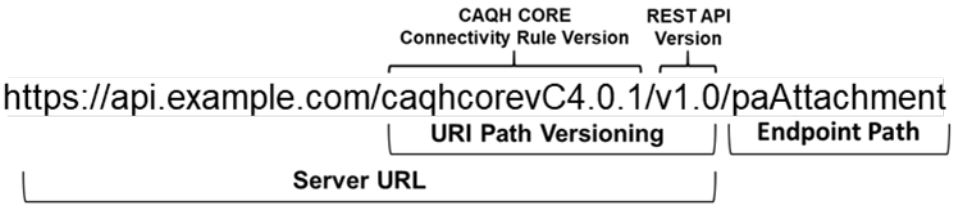
Table 5.3.1 below defines what a normative URI Path format server is required to maintain to support uniform versioning methodologies for REST exchanges.

Table 5.3.1: Specifications for REST API URI Path Versioning

Parameter	URI Path (normative)	Description
CAQH CORE Connectivity Rule vC4.0.0	/caqhcorevC4.0.0	Specifies that the REST API supports CAQH CORE Connectivity Rule vC4.0.0
REST API Version	/v[n[41]]	Specifies the version of REST API supported by the server.

Figure #5.3.1 below provides an informative example showing REST API URI Path Versioning for a Health Care Services Review – Inquiry and Response Endpoint.

Figure #5.3.1: Informative Example for REST API URI Path Versioning



5.3.2. Specifications for REST API URI Path Endpoints for Payload Types (normative)

This rule requires the use of standard naming conventions for REST API endpoints to streamline and support uniform REST implementations. In the context of this rule, a REST API Endpoint refers to one end of a communication channel using REST where each endpoint is the location from which APIs can access the resources necessary to carry out their function.

Table 5.3.2 below defines normative endpoint names servers are required to support for REST exchange of corresponding payload types.

Table 5.3.2: Specifications for REST API URI Path Endpoints for Payload Types

#	Payload Type	Transaction Name	Endpoint Name (normative)
1	X12_270_005010X279A1	Health Care Eligibility Benefit Inquiry and Response	eligibility
2	X12_275_006020X314	Additional Information to Support Health Care Claim or Encounter	claimAttachment
3	X12_275_006020X316	Additional Information to Support Health Care Services Review	paAttachment
4	X12_276_005010X212	Health Care Claim Status Request and Response	claimStatus
5	X12_277_005010X213	Health Care Claim Request for Additional Information	claimStatusRFAI
6	X12_277_006020X313	Health Care Claim Request for Additional Information	claimStatusAI

[41] N refers to a numeric value. Message receivers (servers) are responsible for maintaining versions of their REST APIs.

#	Payload Type	Transaction Name	Endpoint Name (normative)
7	X12_277_005010X214E1_2	Health Care Claim Acknowledgment	claimStatusCA
8	X12_278_005010X217E1_2	Health Care Services Review – Request and Response	servicesReview
9	X12_278_005010X215	Health Care Services Review – Inquiry and Response	servicesReviewIR
10	X12_278_005010X216E2	Health Care Services Review – Notification and Acknowledgment	servicesReviewNA
11	X12_820_005010X218A1	Payroll Deducted and Other Group Premium Payment for Insurance Products	payrollDeducted
12	X12_834_005010X220A1	Benefit Enrollment and Maintenance	benefitEnrollment
13	X12_834_005010X307	Health Insurance Exchange Enrollment	exchangeEnrollment
14	X12_834_005010X318	Plan Member Reporting	memberReporting
15	X12_835_005010X221	Health Care Claim Payment / Advice	remittanceAdvice
16	X12_837_005010X223A1_2	Health Care Claim – Institutional	claimInstitutional
17	X12_837_005010X222A1	Health Care Claim – Professional	claimProfessional
18	X12_837_005010X224A1_2	Health Care Claim – Dental	claimDental
19	X12_999_005010X231A1	Implementation Acknowledgement for Health Care Insurance	ackNack
20	X12_TA1_Response_005010X231A	Interchange Acknowledgment Segment	iaAckNack

This rule is payload agnostic and as such, it supports other non-normative non-X12 Payloads (e.g., HL7_CDA_R2, HL7_C-CDA, .pdf, .txt, .jpeg, etc.). This enables the rule to support a variety of data types and allows capability with existing and emerging standards. As the industry continues to evolve, this rule may be updated to include normative non-X12 Payloads.

5.4. REST HTTP Request Method Requirements

HTTP defines a set of request methods to indicate the desired action to be performed for a given resource. These include those listed below.

- **GET -** requests a representation of the specified resource. Requests using GET should only retrieve data.
- **HEAD -** asks for a response identical to that of a GET request, but without the response body.
- **POST -** is used to submit an entity to the specified resource, often causing a change in state or side effects on the server.
- **PUT -** replaces all current representations of the target resource with the request payload.
- **DELETE -** deletes the specified resource.
- **CONNECT -** establishes a tunnel to the server identified by the target resource.

- **OPTIONS -** is used to describe the communication options for the target resource.
- **TRACE -** performs a message loop-back test along the path to the target resource.
- **PATCH -** is used to apply partial modifications to a resource.

This rule requires the use of the following set of HTTP Methods to indicate the desired action to be performed for a given resource:

Table 5.4: CORE REST Rule Required HTTP Methods

CAQH CORE REST Rule Required HTTP Methods		
Action	**Request URI Path**	**When to Use**
POST	/caqhcore4.0.0/v[n]/[endpoint name[42]]	This method may be used when the client is conducting a real time synchronous or batch asynchronous interaction to submit an X12 request/response transaction with the server, e.g., a 270/271 Health Care Eligibility Benefit Inquiry and Response transaction.
GET	/caqhcore4.0.0/v[n]/[endpoint name]	This method may be used when the client is conducting a batch asynchronous interaction to retrieve a batch X12 with the server, e.g., an 837 Health Care Professional Claim or an 835 Health Care Claim Payment/Advice transaction.

While this rule specifies the use of HTTP Methods POST and GET, entities may choose to use additional HTTP Methods (e.g., PUT, PATCH, DELETE, etc.).

5.5. REST HTTP Metadata, Descriptions, Intended Use and Values

HTTP header fields are components of the header section of request and response messages in the Hypertext Transfer Protocol (HTTP) and define the operating metadata of an HTTP transaction. HTTP headers let the client and the server pass additional information with an HTTP request or response. An HTTP header consists of its case-insensitive name followed by a colon (:), then by its value. The header fields are transmitted after the request line (in case of a request HTTP message) or the response line (in case of a response HTTP message), which is the first line of a message.

This rule specifies metadata that are required to be used for HTTP Requests and HTTP Responses for REST exchanges. These metadata serve as a base of what is required by the CAQH CORE Connectivity REST Rule vC4.0.0. Entities may include additional metadata, as needed.

The REST HTTP Metadata specified in Table 5.5 below pertains to the REST exchanges via the CAQH CORE Connectivity REST Rule vC4.0.0

[42] Endpoint name refers to the REST API URI Path Endpoint for Payload Types defined in Table 4.3.2 of this rule.

Table 5.5 REST HTTP Metadata

Metadata	Description	Requirement	Values	Example
HTTP Request Metadata				
Accept	Media type(s) that is/are acceptable for the response.	Mandatory	application/json	**Accept**: application/json
Authorization	Authorization credentials for HTTP authorization.	Mandatory	A valid OAuth 2.0 token.	**Authorization**: Bearer [Access Token]
Content-Type	The Media type of the body of the request.	Mandatory	application/json	**Content-Type**: application /json
Date	The date and time at which the message was originated.	Mandatory	HTTP-date format[43]	**Date**: Tue, 23 Jun 2020 08:12:31 UTC
Host	The domain name of the server	Mandatory	server domain	**Host: Error! Hyperlink reference not valid.**
HTTP Response Metadata				
Content-Type	The media type of the response content.	Mandatory	application/json	**Content-Type**: application/json
Date	The date and time at which the message was originated.	Mandatory	HTTP-date format	**Date**: Tue, 23 Jun 2020 08:12:31 UTC
Last Modified	The last modified date for the requested object.	Mandatory	HTTP-date format	**Date**: Tue, 23 Jun 2020 08:12:31 UTC
Status Code	The Status line in the HTTP response indicates whether the server responded to the request successfully, or if there was an error.	Situational: When a request is successful or fails, the resource server must responds using the appropriate HTTP status code.	Please see Section 4.3.6.1 for a definition of status and error codes.	**Status**: 401

[43] https://tools.ietf.org/html/rfc7231#section-7.1.1.1

Metadata	Description	Requirement	Values	Example
HTTP Request Metadata				
WWW-Authenticate	Defines the authentication method that should be used to gain access to a resource.	Situational: If the protected resource request does not include authentication credentials or does not contain an access token that enables access to the protected resource, the resource server MUST include the HTTP WWW-Authenticate response header field	HTTP Bearer Authorization Scheme[44]	**WWW-Authenticate:** Bearer

5.6. REST POST Message Structure (Informative Example)

This section shows a generic informative example of a Request and Response message using REST POST, based on the CAQH CORE Connectivity REST Rule vC4.0.0 requirements defined above. They are based on real-world examples but have been updated to use the CAQH CORE-required metadata that is part of this rule.

The Request message structure shown below specifies JSON for payloads transported using REST.

This shows the following components:

1. The HTTP Headers and HTTP Metadata defined as part of the CAQH CORE REST Connectivity Rule vC4.0.0 is shown colored in blue.

2. An example of a resource specific metadata is shown colored in green.

3. An example X12_275_006020X316 payload is shown colored in pink.

4. For this example, the BDS*ASC*551280* will be a placeholder for the actual Attachment, in this sample X12 payload and is shown colored in yellow.

[44] https://www.iana.org/assignments/http-authschemes/http-authschemes.xhtml

```
POST /caqhcorev4.0.0/v1.0/paAttachment HTTP/1.1
“Accept”:[“application/json ”]
.“Authorization”:[“Bearer [Access Token]”]
,”Content-Type”:[“application /json”]
.”Date”:[“Tue, 23 Jun 2020 08:12:31 UTC”]
.“Host”:[“https://www.acmehealthplan.com”]

{
  "ResourceType": [
    "Endpoint"
  ],
  "X12_ID": [
    "paAttachment"
  ],
  "X12_identifier": [
    {
      "location": "http://core.org/endpoint-identifier",
      "type": "X12_275_006020X316"
    }
  ],
  "Meta": "",
  "Status": [
    "active"
  ],
  "Version": [
    1
  ],
  "ConnectionType": [
    {
      "location": "http://www.core.org/endpoint-connection-type",
      "type": "core-rest"
    }
  ],
  "Name": [
    "CAQH CORE"
  ],
  "ManagingOrganization": [
    "Organization/caqhcore"
  ],
  "Contact": [
    "admin@caqhcore.org"
  ],
  "Period": [
    "9/1/2020"
  ],
  "PayloadType": [
    {
      "location": "http://core.org/payload-types",
      "type": "X12_278_Request_005010X217E1_2"
    }
  ],
  "ProcessingMode": [
    "RealTime"
  ],
  "PayloadLength": [
    "1551254"
```

```
  ],
  "PayloadMimeType": [
    "application/edi-x12"
  ],
  "PayloadID": [
    "1234ABCDEF567890"
  ],
  "Timestamp": [
    "20200803_0655443322"
  ],
  "SenderID": [
    "HospitalA"
  ],
  "ReceiverID": [
    "PayerB"
  ],
  "CORERuleVersion": [
    "C4.0.0"
  ],
  "Checksum": [
    "0987654321ABXDEF"
  ],
  "SecurityMode": [
    {
      "type": "OAuth",
      "token": "0987654321ABCDEF1234567890"
    }
  ],
  "Data": [
    {
      "X12_275_006020X316": [
        "ISA*00*Corerest00*00*Security I*ZZ*Interchange Sen*ZZ*Interchange
Rec*181212*1037*^*00602*000031033*0*T*:^ GS*HI*Sample Sen*Sample
Rec*20201212*1037*123456*X*006020X316^ ST*275*0001*006020X316^ BGN*02*244579*20201212*1037^
NM1*ACV*2*PCPOffice******46*1245233261*67*1P^ NM1*40*2*PrimaryPayer******PI*1245233261*67*PR^
NM1*IL*1*ADAMS*AMANDA****MI*PPMI00001253^ LX*1^ TRN*1*ACN1331^ DTP*368*D8*20201212^
CAT*AE*IA^ OOI*1*47*ATTACHMENT BDS*ASC*551280*<.....><Note: this data element contains a string of
octets which can assume any binary data pattern. The maximum length depends on the data value entered prior
data element; in this example the value is 551280.>^ SE*12*0001^ GE*1*123456^ IEA*1*000031033^"
      ]
    }
  ]
}
```

This example depicts a REST response acknowledging the initial request. The Response message structure shown below specifies JSON for payloads transported using REST.

This shows the following components:

1. The HTTP Headers and HTTP Metadata defined as part of the CAQH CORE REST Connectivity Rule vC4.0.0 is shown colored in blue.

2. An example of a resource specific metadata is shown colored in green.

3. An example X12_999_005010X231A1payload is shown colored in pink.

```
”Content-Type”:[“application /json”]
.”Date”:[“Tue, 23 Jun 2020 08:12:31 UTC”]
.“Status:[202]
,”WWW-Authenticate”:[”Bearer”]

{
 "ResourceType": [
   "Endpoint"
 ],
 "X12_ID": [
   "ackNack"
 ],
"X12_identifier": [
   {
     "location": "http://core.org/endpoint-identifier",
     "type": "X12_999_005010X231A1"
   }
 ],
 "Meta": "",
 "Status": [
   "active"
 ],
"Version": [
   1
 ],
 "ConnectionType": [
   {
     "location": "http://www.core.org/endpoint-connection-type",
     "type": "core-rest"
   }
 ],
 "Name": [
   "CAQH CORE"
 ],
 "ManagingOrganization": [
   "Organization/caqhcore"
 ],
 "Contact": [
   "admin@caqhcore.org"
 ],
 "Period": [
   "9/1/2020"
 ],
```

```
  "PayloadType": [
    {
      "location": "http://core.org/payload-types",
      "type": "X12_BatchReceiptConfirmation"
    }
  ],
    "ProcessingMode": [
    "Batch"
  ],
  "PayloadLength": [
    "155"
  ],
"PayloadMimeType": [
    "application/edi-x12"
  ],
  "PayloadID": [
    "f81d4fae-7dec-11d0-a765-00a0c91e6bf6"
  ],
  "TimeStamp": [
    "2007-08-30T10:20:34Z"
  ],
  "SenderID": [
    "PayerB"
  ],
  "ReceiverID": [
    "HospitalA"
  ],
  "CORERuleVersion": [
    "C4.0.0"
  ],
"checksum": [
    "0987654321ABXDEF"
  ],
  "ErrorCode": [
    "Success"
  ],
  "ErrorMessage": [
    "This is an Error"
  ],
  "Data": [
    {
      "X12_999_005010X231A1": [
        "ISA~00~Authorizat~00~Security I~01~Interchange Sen~01~Interchange
Rec~080701~1240~*~00501~000000001~0~I~^GS~FA~Application Sen~Application
Rec~20080701~12405600~1~T~005010X231A1~999~0001~005010X231A1^AK1~HI~123456~006020X316^AK
2~275~001^IK5~A^AK9~A~1~1~1^SE~5~0001^GE~1IEA~1~000000001"
      ]
    }
  ]
}
```

5.7. Publication of Entity-Specific Connectivity Companion Document

Servers must publish detailed specifications in a Connectivity Companion Document on the entity's web site. CAQH 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 CAQH CORE rules.

- Details on supported transactions and processing modes (e.g., Real Time, Batch transactions).
- Details about the entity's X12 Interchange, e.g., will an interchange contain multiple functional groups; will the TA1 be in its own interchange without any functional group(s).
- CAQH CORE Connectivity Rule Version for REST Connectivity.
- Details about versioning of REST API for REST Connectivity (e.g., dates, version number, etc.).
- List of URI Paths for API Endpoints.
- HTTP Requests Methods supported by REST API.
- Details about HTTP Metadata supported by REST API
- Production and Testing URLs for Synchronous Real Time and Asynchronous Batch transactions.
- Maximum size of X12 Interchange payload for Asynchronous Batch Processing Mode that can be received by a Server.
- Authorization policies using OAuth 2.0, e.g., how to obtain an access token from a receiver (server) to make a request on a resource.
- 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 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 TLS 1.2 or higher and the algorithm (e.g., SHA-2) that is expected for Checksum element.

6. CAQH CORE Safe Harbor

This rule specifies a "Safe Harbor" that any stakeholder can be assured will be supported by any HIPAA-covered entity and its agent. This rule further specifies the connectivity method that all HIPAA-covered entities and 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 CAQH 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 CAQH CORE Connectivity Rule vC4.0.0. When a HIPAA-covered entity and 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 CAQH CORE Connectivity Rule vC4.0.0 requirements. (See §4.4.3.1)

This CAQH CORE Connectivity Rule vC4.0.0 is the CAQH CORE Safe Harbor connectivity method that a HIPAA-covered entity and its agent MUST use if requested by a trading partner. If the HIPAA-covered entity and 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 and 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.

The sections below specify the conformance requirements for stakeholders that can be CORE-certified, including Health Plans, Clearinghouses, Providers, Provider Vendors and Health Plan Vendors.

6.1.1. Health Plans and Health Plan Vendors

Health Plans and Health Plan Vendors (servers) must implement capability to support CAQH CORE Connectivity SOAP (§4) and REST (§5) Rule vC4.0.0 requirements.

6.1.2. Clearinghouses, Health Information Exchanges, and Other Intermediaries

Intermediaries, including Clearinghouses, Switches, and Health Information Exchanges, act as both client and server. The server portion of Clearinghouses/Switches/Health Information Exchanges must implement the capability to support CAQH CORE Connectivity SOAP (§4) and REST (§5) Rule vC4.0.0 requirements.

6.1.3. Providers and Provider Vendors

Providers and Provider Vendors (clients) must implement capability to support either CAQH CORE Connectivity SOAP (§4) or REST (§5) Rule vC4.0.0 requirements. If a Provider or a Provider Vendor implement a server, then it must implement capability to support CAQH CORE Connectivity SOAP (§4) and REST (§5) Rule vC4.0.0 requirements.

7. Conformance Requirements

Conformance with this CAQH CORE Operating Rule can be voluntarily demonstrated and certified through successful completion of the approved CAQH CORE Certification Test Suites with a third party CAQH 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 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 ACA, HIPAA mandates a certification process for HIPAA-covered health plans only, under which HIPAA-covered 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 HIPAA-covered 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))

8. Appendix

8.1. References

Note: These were used for rule creation as well as to create the analysis artifacts as part of CAQH CORE Connectivity.

Table 7.1		
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	https://www.ietf.org/rfc/rfc3986.txt
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
Internet Engineering Task Force (IETF)	OAuth 2.0 Authorization Framework	https://tools.ietf.org/html/rfc6749
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

Table 7.1		
Author	**Document Name**	**Location**
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

8.2. Abbreviations and Definitions Used in this Rule

Table 7.2	
Term or Concept	**Definition**
X12 Interchange	An X12 Interchange is a graphic character string structured using delimited, tagged data concepts. An X12 Interchange begins with an Interchange Control Header segment: Segment ID = ISA and ends with an Interchange Control Trailer segment: Segment ID – IEA. An X12 Interchange may be composed of one or more Functional Groups (GS/GE Control Segments). An X12 Functional Group is composed of one or more Transaction Sets (ST/SE Control Segments). An 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 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.

Table 7.2	
Term or Concept	**Definition**
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)[45] 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[46] 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 X12 Interchange containing one Functional Group containing one X12 transaction set consisting of more than one business transaction.
Client	An entity that sends/relays a message to a Server.
CAQH 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 and their agents.
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.[47] 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.[48]

[45] CORE Phase I Glossary Definitions. http://www.caqh.org/sites/default/files/core/phase-i/reference/PIGlossary.pdf
[46] Ibid.
[47] http://www.atis.org/glossary/definition.aspx?id=7853 ATIS (Alliance for Telecommunications Industry Solutions http://www.atis.org/about/index.asp.
[48] http://en.wikipedia.org/wiki/Extensibility.

Table 7.2	
Term or Concept	**Definition**
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.[49] 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[50].

[49] Adapted from http://engineers.ihs.com/document/abstract/AQSBFBAAAAAAAAAA ANSI Information Technology – Vocabulary – Part 1: Fundamental Terms.
[50] HIMSS Position Statement: Adoption of HITSP Interoperability Specifications July 2007.

Table 7.2	
Term or Concept	**Definition**
Interoperability Specification[51]	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 X12 Interchange, each of which may contain one or more Functional Groups, each of which may contain one or more 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/).

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

Table 7.2	
Term or Concept	**Definition**
Normative	In standards terminology, "normative" means "considered to be a prescriptive part of the standard" [Wikipedia].
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[52]	"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).[53]
Performance	According to the CAQH CORE Connectivity Rule vC1.1.0, 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 CAQH CORE Eligibility and Benefits Rules and Policies for Real time [54] and Batch [55] 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 CAQH CORE Eligibility and Benefits Rules and Policies for system availability [56].

[52] International Telecommunication Union – Open Standards Definition. http://www.itu.int/ITU-T/othergroups/ipr-adhoc/openstandards.html.
[53] SearchSecurity.com. http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci214475,00.html.
[54] CAQH CORE Eligibility & Benefits (270/271) Data Content Rule vEB1.0
[55] Ibid
[56] Ibid

Table 7.2	
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.[57] 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.[58] 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.[59] 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)	Real Time Mode [60] 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.

[57] CAQH CORE Eligibility & Benefits (270/271) Infrastructure Rule vEB.1.0 (formerly the CORE Phase I 156: Eligibility and Benefits Real Time Response Time Rule; and CORE Phase I 155: Eligibility and Benefits Batch Response Time Rule.)
[58] http://www.cs.washington.edu/homes/lazowska/qsp/Contents.pdf Quantitative System Performance, Chapter 5.2.1. Transaction Workloads (Page 72).
[59] http://www.atis.org/tg2k/ and search "Overhead Information" ATIS (Alliance for Telecommunications Industry Solutions http://www.atis.org/about/index.asp.
[60] Ibid.

Table 7.2	
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.[61] 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.[62]
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.
Authentication	X.509 Certificate based Authentication over SSL or TLS, described in Sub-section "Authentication Handling."
Authorization	OAuth Token based Authorization over SSL or TLS, described in Sub-section "Authorization 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)[63] 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).

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

8.3. Sequence Diagrams for SOAP Rule (§4)

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 X12 999 or X12 TA1 payloads).

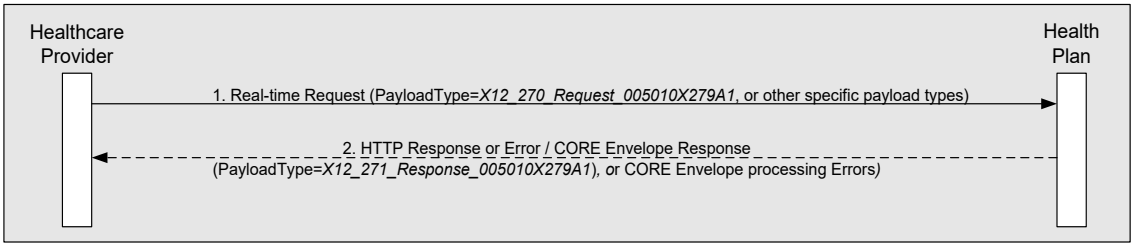
8.3.1. Real Time Interaction

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 Eligibility Benefit Inquiry and Response (X12 v5010 270/271)

The UML sequence diagram below shows a *Health Care Eligibility Benefit Inquiry 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.

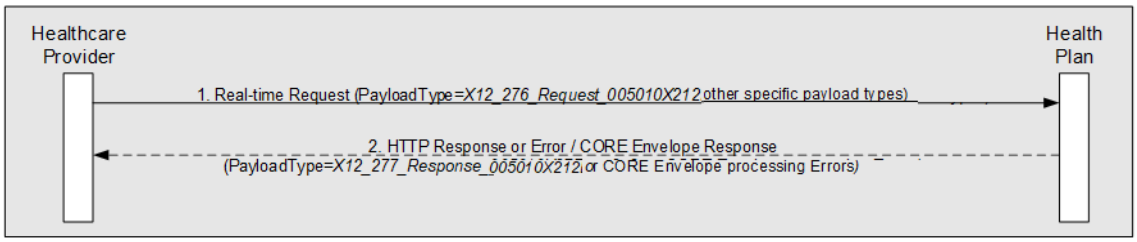


The following describes the typical Real-time interaction as shown in the above diagram.

Message Sequence	Description
1	Healthcare Provider submits a Real-time request to the Health Plan, using payload type as X12_270_Request_005010X279A1
2	Health Plan responds (synchronously to request message 1) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (Payload type is X12_271_Response_005010X279A1 or error).

Example 2: Health Care Claim Status Request and Response (X12 v5010 276/277)

The UML sequence diagram below shows a *Health Care Claim Status Request 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.

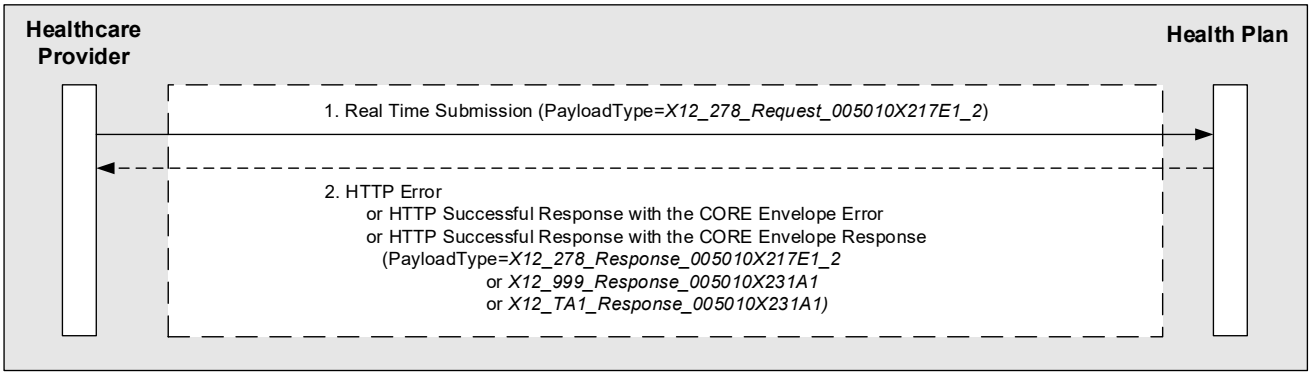


The following describes the typical Real-time interaction as shown in the above diagram.

Message Sequence	Description
1	Healthcare Provider submits a Real-time request to the Health Plan, using payload type as X12_276_Request_005010X276212
2	Health Plan responds (synchronously to request message 1) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (Payload type is X12_277_Response_005010X277212 or error).

Example 3: Health Care Services Review – Request for Review and Response (X12 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.

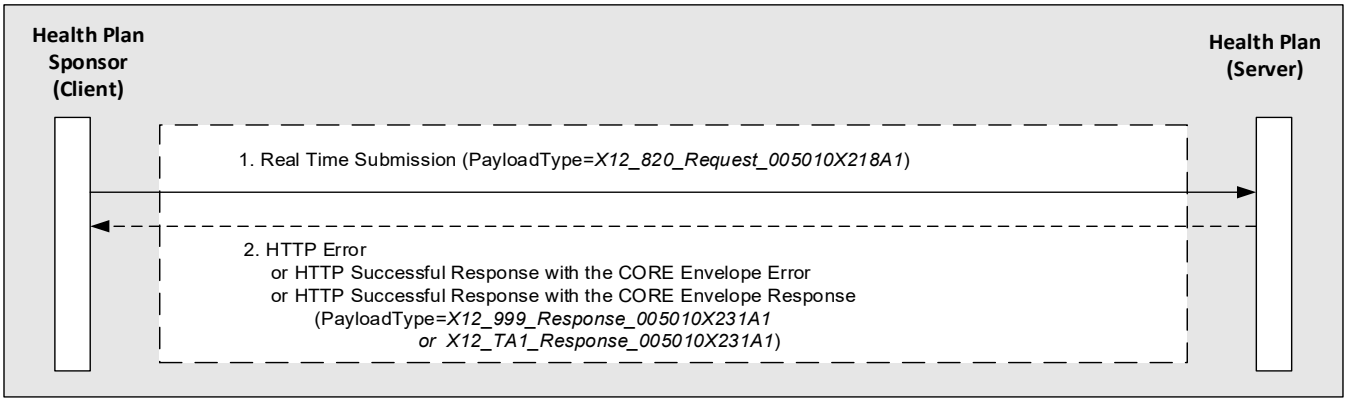


The requester of a Real Time response expects one and only one response on the payload; for example, in the above message interaction, the response payload can only be an X12 v5010 278, or an X12 v5010 999 or an X12 TA1. The following describes the Real Time interaction as shown in the above diagram.

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
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 4: Payroll Deducted and Other Group Premium Payment for Insurance Products (X12 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.

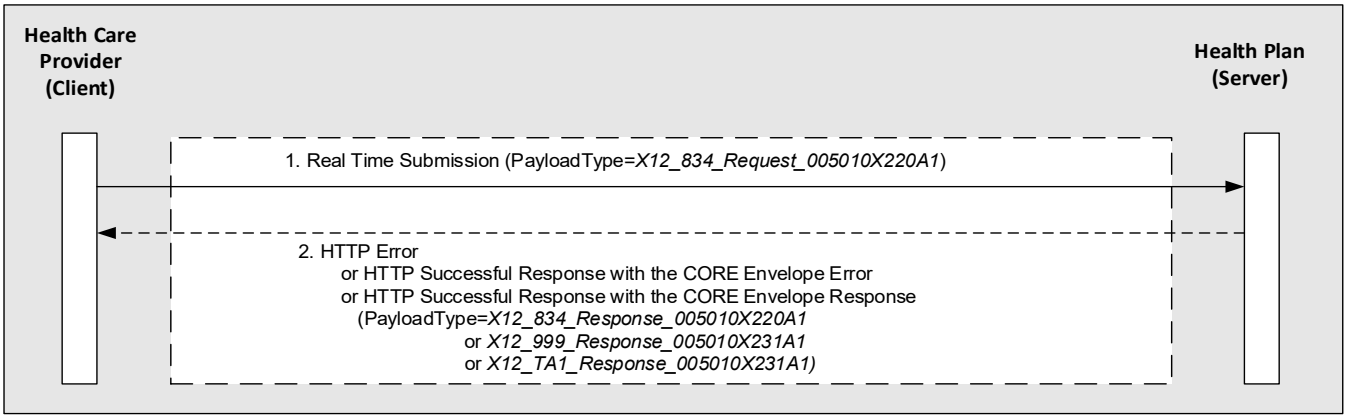


The requester of a Real Time response expects one and only one response on the payload; for example, in the above message interaction, the response payload can only be an X12 v5010 999, or an X12 TA1. The following describes the Real Time interaction as shown in the above diagram.

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
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 5: Benefit Enrollment and Maintenance (X12 v5010 834)

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

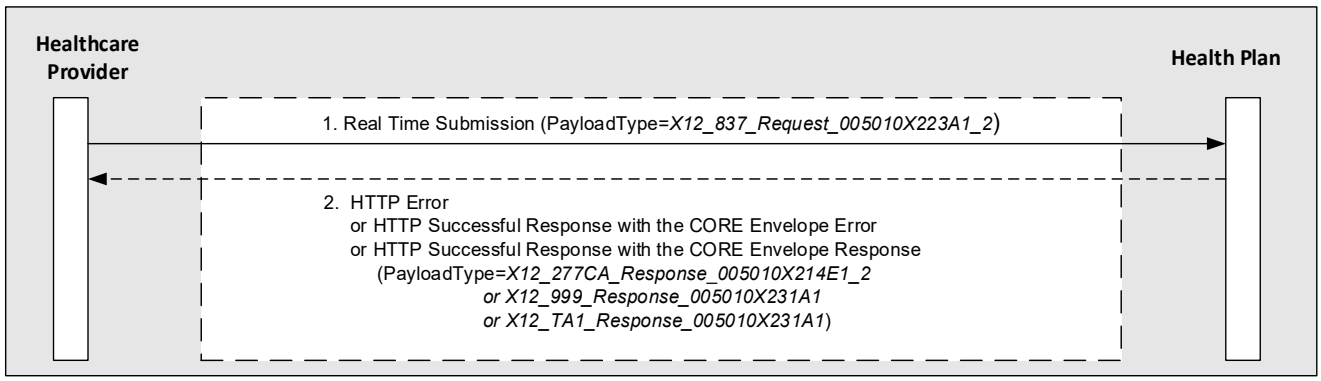


The requester of a Real Time response expects one and only one response on the payload; for example, in the above message interaction, the response payload can only be only an X12 v5010 834, X12 v5010 999, or an X12 TA1. The following describes the Real Time interaction as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Health Care Provider submits a Real Time request to a Health Plan, using PayloadType=*X12_834_Request_005010X220A1*.	Benefit Enrollment and Maintenance
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_834_Response_005010X220A1* *or X12_999_Response_005010X231A1* *or X12_TA1_Response_005010X231A1*)	Benefit Enrollment and Maintenance

Example 6: Healthcare Claim (X12 v5010 837 Claim)

The UML sequence diagram below shows an Institutional *Healthcare Claim* Real Time transaction between a HIPAA-covered Healthcare Provider (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 X12 v5010 277CA, or an X12 v5010 999, or an X12 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_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

8.3.2. 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)

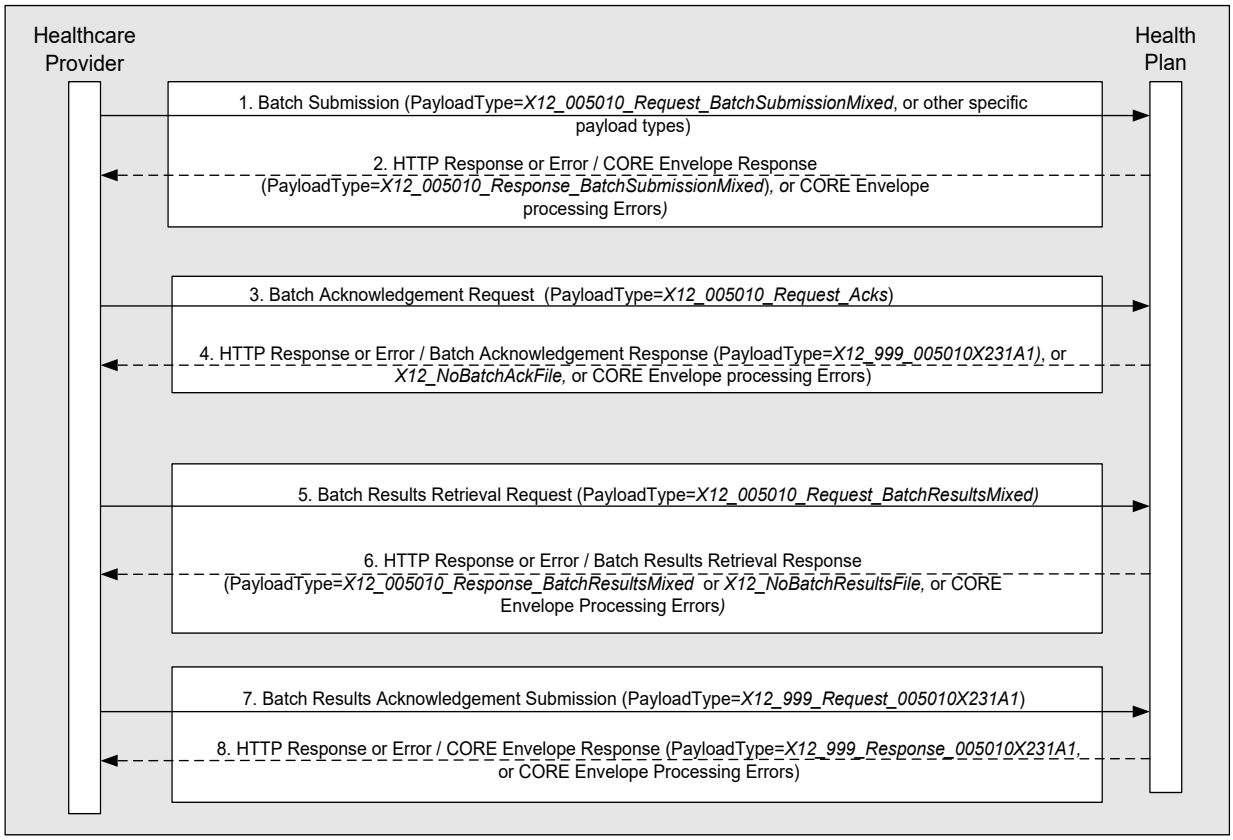
The Batch interactions can be conducted using specific payload types as shown in 7.3.2.1 or with Mixed Payload types as show in 7.3.2.2.

8.3.2.1. 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: Batch Submission:

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



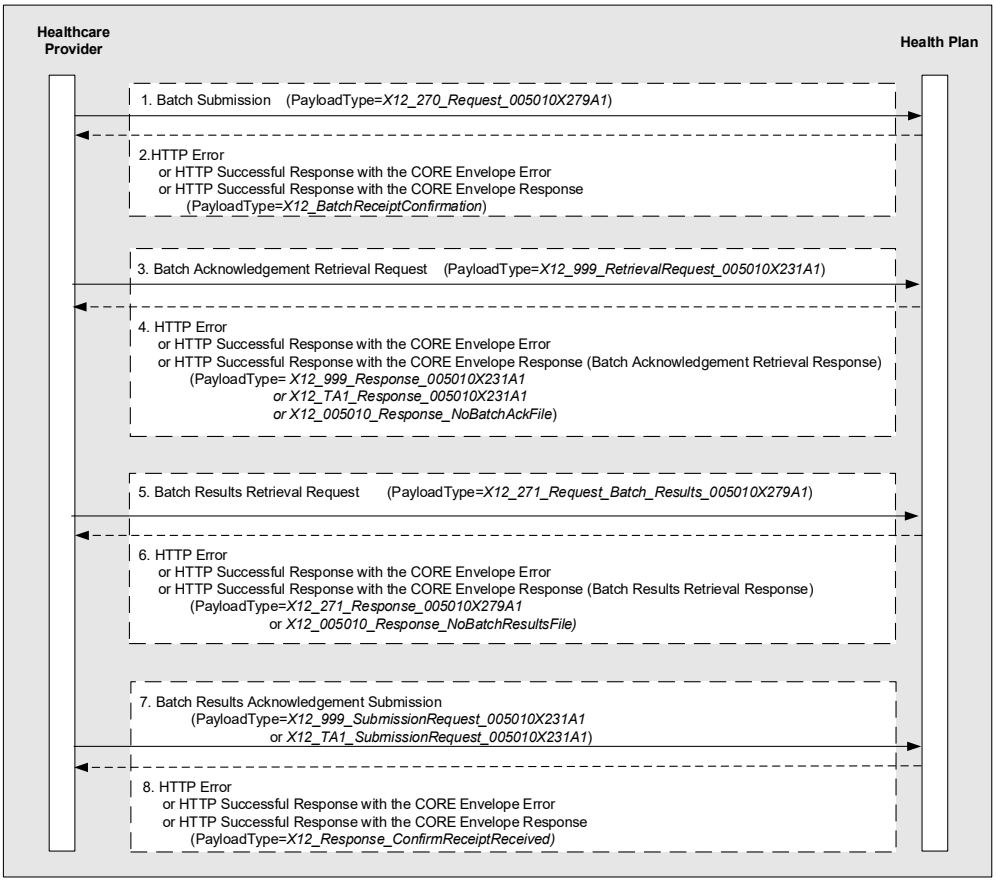
The following describes the typical Batch interaction as shown in the above diagram.

Message Sequence	Description
1	Healthcare Provider submits a Batch of requests to the Health Plan, using payload type as BatchSubmissionMixed (e.g., payload type=X12_005010_Request_BatchSubmissionMixed), or one of the specific payload types.
2	Health Plan responds (synchronously to request message 1) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), indicating that the Batch was received (e.g., payload type = X12_005010_Response_BatchSubmissionMixed) and the CORE envelope was processed (with or without errors).
3	Healthcare Provider sends a Request to the Health Plan to solicit the acknowledgement (X12 v5010 999 or TA1) for the Batch file that was just submitted.
4	Health Plan responds (synchronously to request message 3) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), with the X12 v5010 999 or an X12_TA1 acknowledgement. If no v5010 999 or TA1 is ready for pickup, Health Plan sends a CORE Envelope with payload type set to X12_NoBatchAckFile.
5	Healthcare Provider sends a Request to the Health Plan to solicit the Results for the Batch file that was submitted in message sequence 1.

Message Sequence	Description
6	Health Plan responds (synchronously to request message 5) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), with the payload type set to X12_BatchResults, and sends the result file as payload. If no results file is ready for pickup, Health Plan sends a CORE Envelope with payload type set to X12_NoResultsFile.
7	Healthcare Provider submits the acknowledgement (payload type X12_999_Request_005010X231 or X12_TA1) to the Health Plan
8	Health Plan responds (synchronously to request message 7) to the request either with an HTTP level error, or an HTTP successful response accompanied by a CORE envelope level response (or error), indicating that the Batch results acknowledgement was received (payload type =X12_999_Response_005010X231) and the CORE envelope was processed (with or without errors).

Example 2: Health Care Eligibility Benefit Inquiry and Response (X12 v5010 270/271):

The UML sequence diagram below shows a typical Batch Interaction between a HIPAA-covered Healthcare Provider, and a HIPAA-covered Health Plan specifically for X12 v5010 270/271 batch payloads.



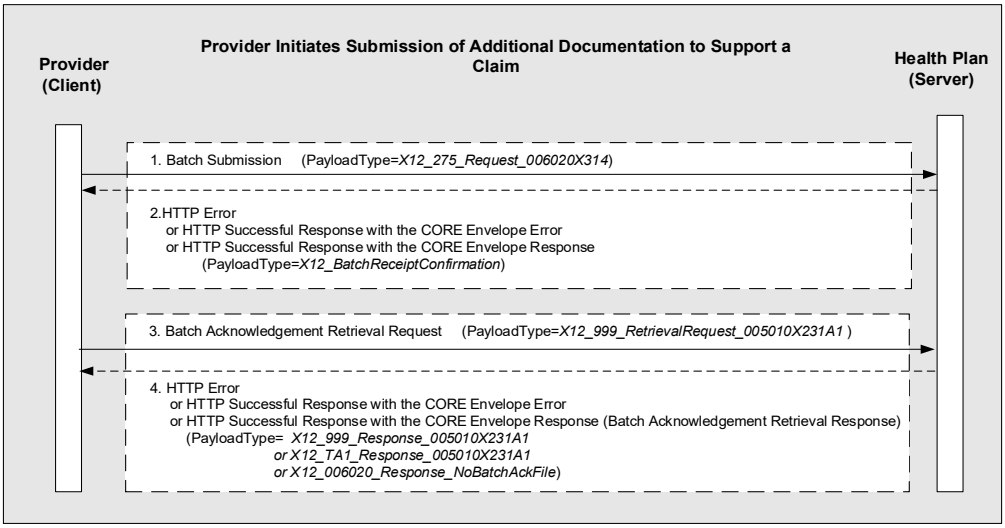
The following describes the Batch interaction for X12 v5010 270/271 batch payloads as shown in the above diagram.

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_270_Request_005010X279A1*.	Health Care Eligibility Benefits Inquiry and Response
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_005010X231A1* *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_Request_Batch_Results_271*.	Health Care Eligibility Benefits Inquiry and 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_271_Response_005010X279A1* or *X12_005010_Response_NoBatchResultsFile)*	Health Care Eligibility Benefits Inquiry and 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 CAQH CORE Connectivity Rule vC1.1.0 and CAQH CORE Connectivity Rule vC2.2.0.	Implementation Acknowledgement Submission or Interchange 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

Example 3: Additional Information to Support a Health Care Claim or Encounter (X12 v6020 275):

The UML sequence diagram below shows a typical Batch Interaction between a HIPAA-covered Healthcare Provider, and a HIPAA-covered Health Plan specifically for X12 v6020 275 batch payloads.



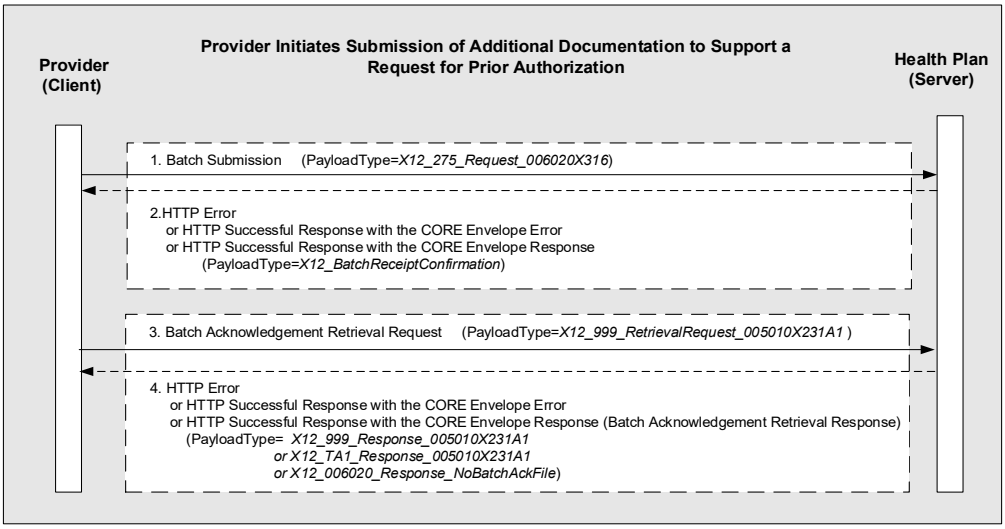
The following describes the *Additional Documentation (Attachment)* transaction using the *Generic Push* interaction as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Healthcare Provider submits a Batch of Attachments to a Health Plan, using PayloadType = *X12_275_Request_006020X314 for a health care claim*	Additional Information to Support a Health Care Claim or Encounter
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_06020X290*) 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 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_06020X290* *or X12_TA1_Response_06020X290* *or X12_006020_Response_NoBatchAckFile)*	Implementation Acknowledgement Retrieval
5.	A Health Plan responds to the initial data content processing of an Attachment using Payload Type= *X12_824_Response_00620X257*	Application Reporting for Insurance
6.	A Health Plan sends a Request to the Provider to solicit the acknowledgement (*X12_999_RetrievalRequest_06020X290)*) for the application advice file that was just submitted.	Implementation Acknowledgement Retrieval
7.	A Provider responds (synchronously to request message 6) 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_06020X290* *or X12_TA1_Response_06020X290* *or X12_006020_Response_NoBatchAckFile)*	Implementation Acknowledgement Retrieval

Example 4: Additional Information to Support a Prior Authorization Request (X12 v6020 275):

The UML sequence diagram below shows a typical Batch Interaction between a HIPAA-covered Healthcare Provider, and a HIPAA-covered Health Plan specifically for X12 v6020 275 batch payloads.

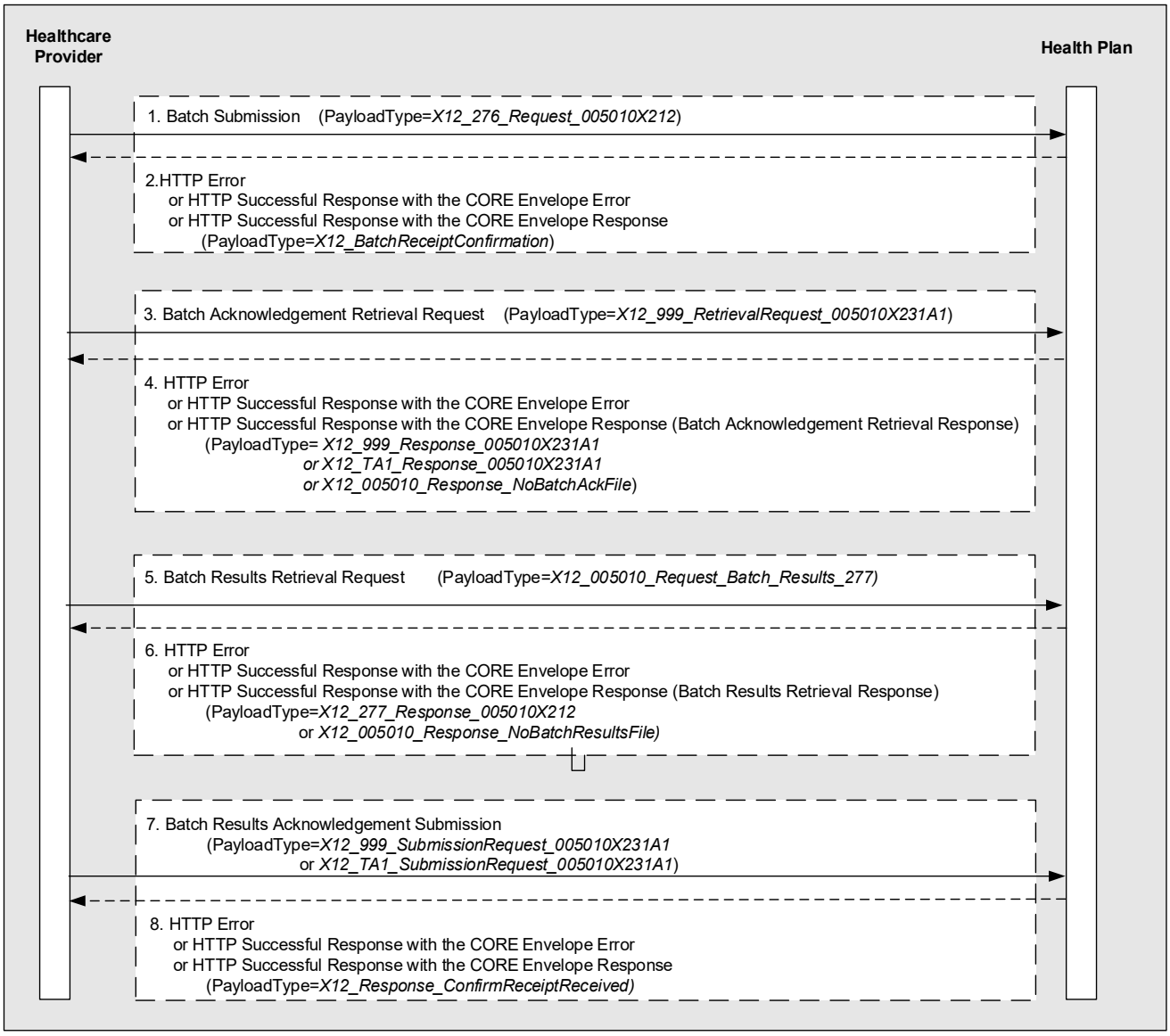

Provider Initiates Submission of Additional Documentation to Support a Request for Prior Authorization
Provider (Client)
Health Plan (Server)
1. Batch Submission (PayloadType=X12_275_Request_006020X316)
2.HTTP Error
or HTTP Successful Response with the CORE Envelope Error
or HTTP Successful Response with the CORE Envelope Response
(PayloadType=X12_BatchReceiptConfirmation)
3. Batch Acknowledgement Retrieval Request (PayloadType=X12_999_RetrievalRequest_005010X231A1)
4. 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_005010X231A1
or X12_006020_Response_NoBatchAckFile)

The following describes the *Additional Documentation (Attachment)* 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	A Healthcare Provider submits a Batch of Attachments to a Health Plan, using PayloadType = *X12_275_Request_006020X316 for a prior authorization*	Additional Information to Support a Health Care Services Review
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_006020_Response_NoBatchAckFile)*	Implementation Acknowledgement Retrieval
5.	A Health Plan responds to the initial data content processing of an Attachment using Payload Type= *X12_824_Response_00620X257*	Application Reporting for Insurance
6.	A Health Plan sends a Request to the Provider to solicit the acknowledgement (*X12_999_RetrievalRequest_06020X290)*) for the application advice file that was just submitted.	Implementation Acknowledgement Retrieval
7.	A Provider responds (synchronously to request message 6) 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_06020X290* *or X12_TA1_Response_06020X290* *or X12_006020_Response_NoBatchAckFile)*	Implementation Acknowledgement Retrieval

Example 5: Health Care Claim Status Request and Response (X12 v5010 276/277):

The UML sequence diagram below shows a typical Batch Interaction between a HIPAA-covered Healthcare Provider, and a HIPAA-covered Health Plan specifically for X12 v5010 276/277 batch payloads.



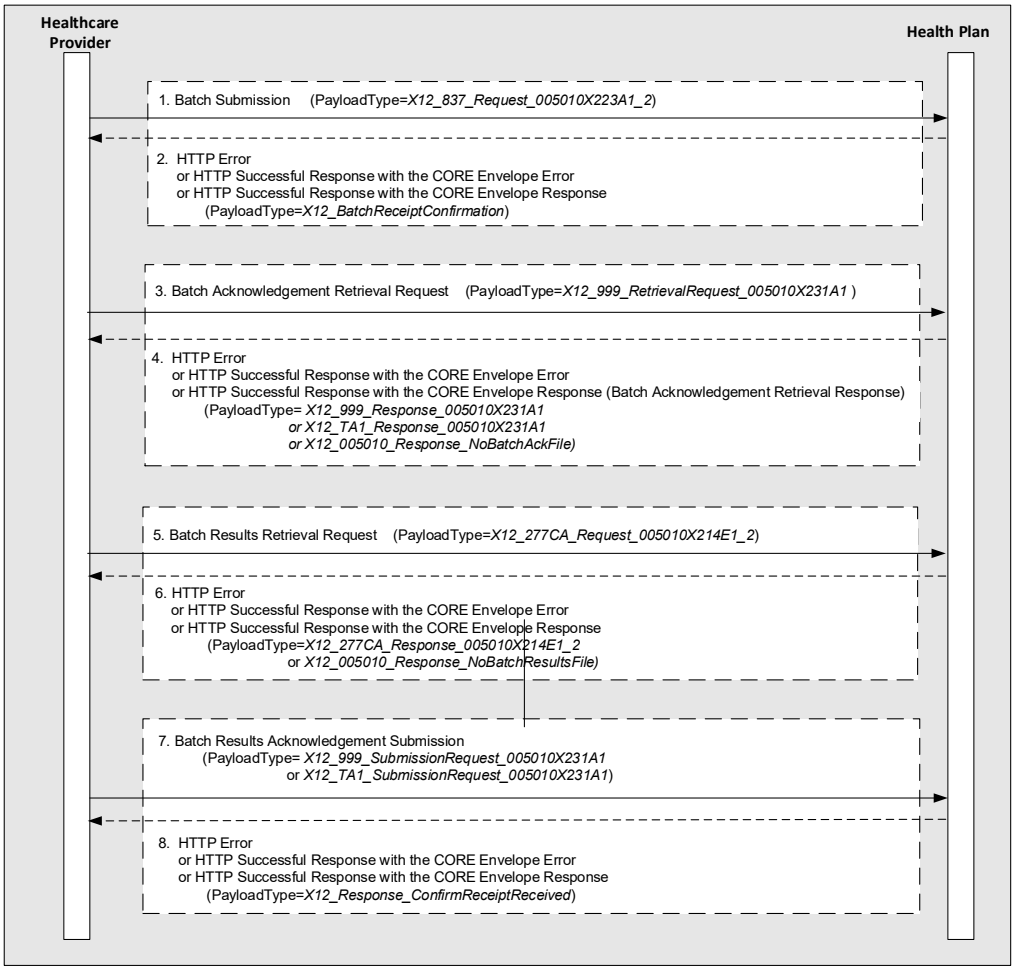
The following describes the Batch interaction for X12 v5010 276/277 batch payloads as shown in the above diagram.

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_276_Request_005010X212*.	Health Care Claim Status Request and Response
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_005010X231A1* *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_005010_Request_Batch_Results_277*.	Health Care Claim Status Request and 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_277_Response_005010X212* or *X12_005010_Response_NoBatchResultsFile)*	Health Care Claim Status Request and Response

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 (PayloadType= *X12_999_SubmissionRequest_005010X231A1*, or *X12_TA1_SubmissionRequest_00501X231A1)* to a Health Plan. This acknowledgment submission is required by CAQH CORE Connectivity Rule vC1.1.0 and CAQH CORE Connectivity Rule vC2.2.0.	Implementation Acknowledgement Submission or Interchange Acknowledgement Submission
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

Example 6: Health Care Claim (X12 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 X12 v5010 837 batch payloads.



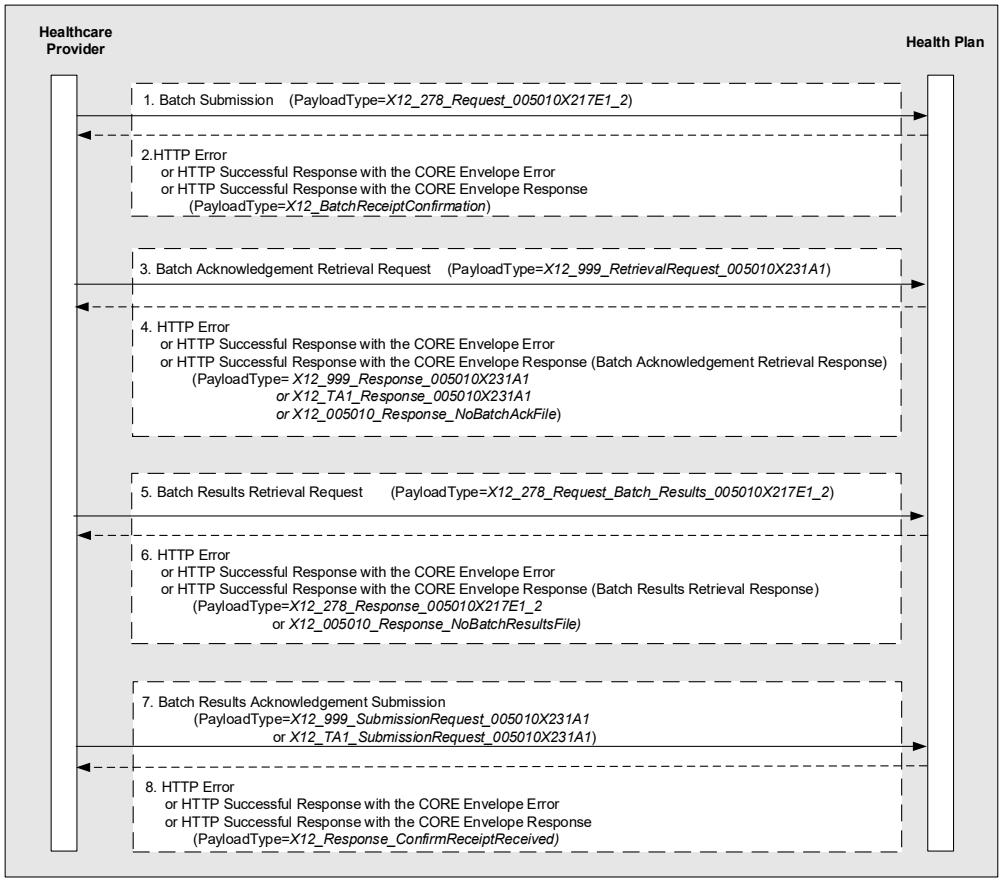
The following describes the Batch interaction for X12 v5010 837 batch payloads as shown in the above diagram.

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_005010X231A1* *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 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 7: Health Care Services Review – Request for Review & Response (X12 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 X12 v5010 278 batch payloads.



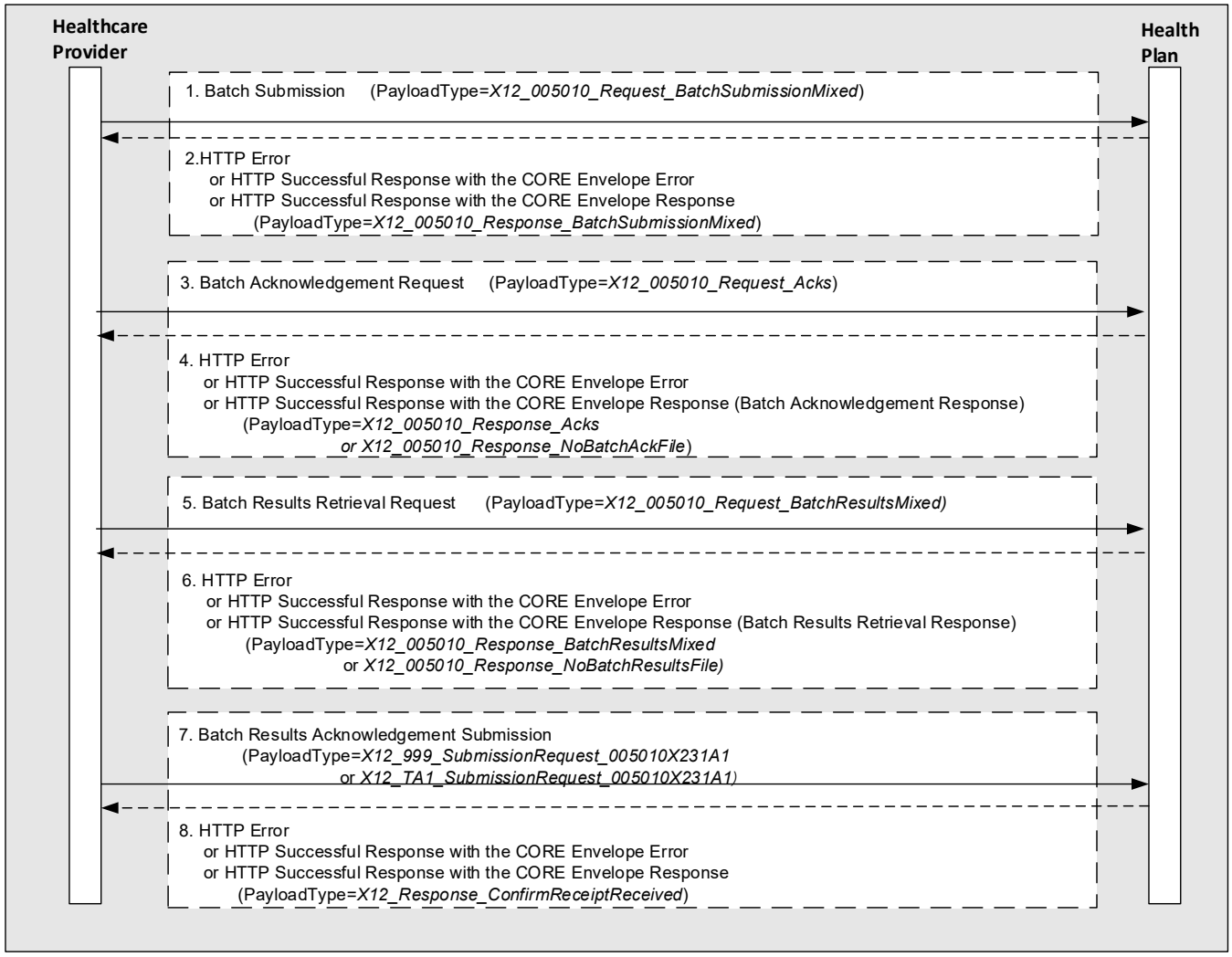
The following describes the Batch interaction for X12 v5010 278 batch payloads as shown in the above diagram.

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
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

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
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_005010X231A1* *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 CAQH CORE Connectivity Rule vC1.1.0 and CAQH CORE Connectivity Rule vC2.2.0.	Implementation Acknowledgement Submission
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

8.3.2.2. Batch Interaction for Mixed Payload Types

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


Healthcare Provider
Health Plan
1. Batch Submission (PayloadType=X12_005010_Request_BatchSubmissionMixed)
2.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)
3. Batch Acknowledgement Request (PayloadType=X12_005010_Request_Acks)
4. 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)
5. Batch Results Retrieval Request (PayloadType=X12_005010_Request_BatchResultsMixed)
6. 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)
7. Batch Results Acknowledgement Submission
(PayloadType=X12_999_SubmissionRequest_005010X231A1
or X12_TA1_SubmissionRequest_005010X231A1)
8. HTTP Error
or HTTP Successful Response with the CORE Envelope Error
or HTTP Successful Response with the CORE Envelope Response
(PayloadType=X12_Response_ConfirmReceiptReceived)

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 (X12 v5010 999 or X12 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 CAQH CORE Connectivity Rule vC1.1.0 and CAQH CORE Connectivity Rule vC2.2.0.	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

8.3.3. 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.

Example transactions are shown in the following sub-sections:

a) Health Care Claim Payment/Advice (X12 v5010 835)
b) Benefit Enrollment and Maintenance (X12 v5010X220 834)
c) Payroll Deducted and Other Group Premium Payment for Insurance Products (X12 v5010 820)
d) Plan Member Reporting (X12 v5010X318 834)

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.

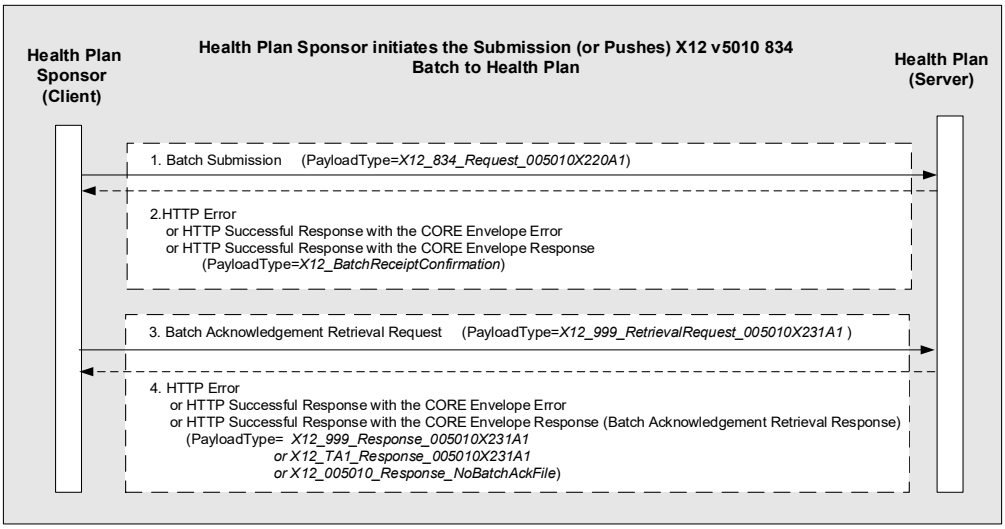
8.3.3.1. Generic Push

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.

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

Example 8: Benefit Enrollment and Maintenance (X12 v5010 834)

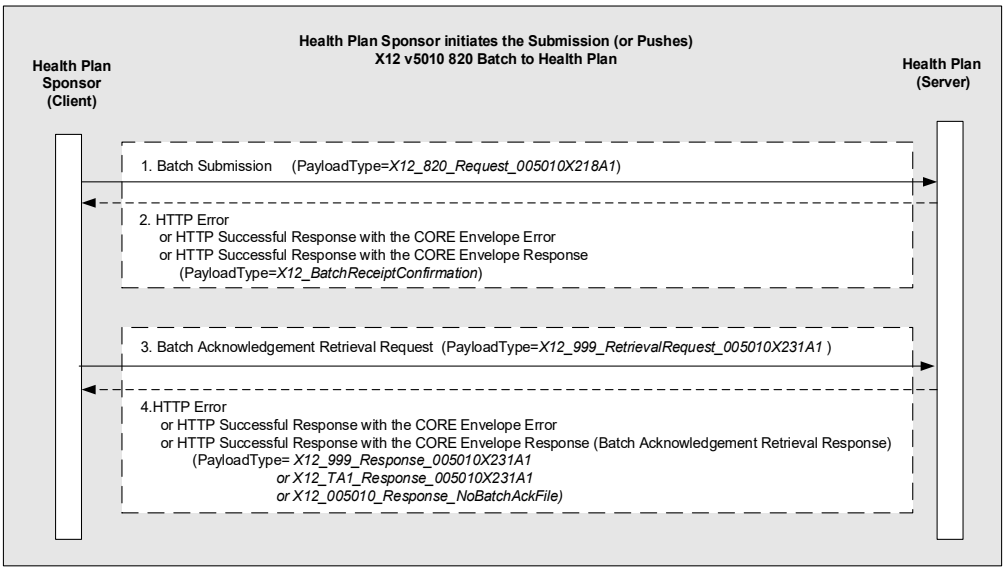


The following describes the *Benefit Enrollment and Maintenance* transaction using the *Generic Push* interaction as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	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 (X12 v5010 999 or X12 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_005010X231A1* *or X12_005010_Response_NoBatchAckFile*)	Benefit Enrollment and Maintenance

Example 9: Payroll Deducted and Other Group Premium Payment for Insurance Products (X12 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

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
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 (X12 v5010 999 or X12 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_005010X231A1* *or X12_005010_Response_NoBatchAckFile)*	Implementation Acknowledgement Retrieval (Response)

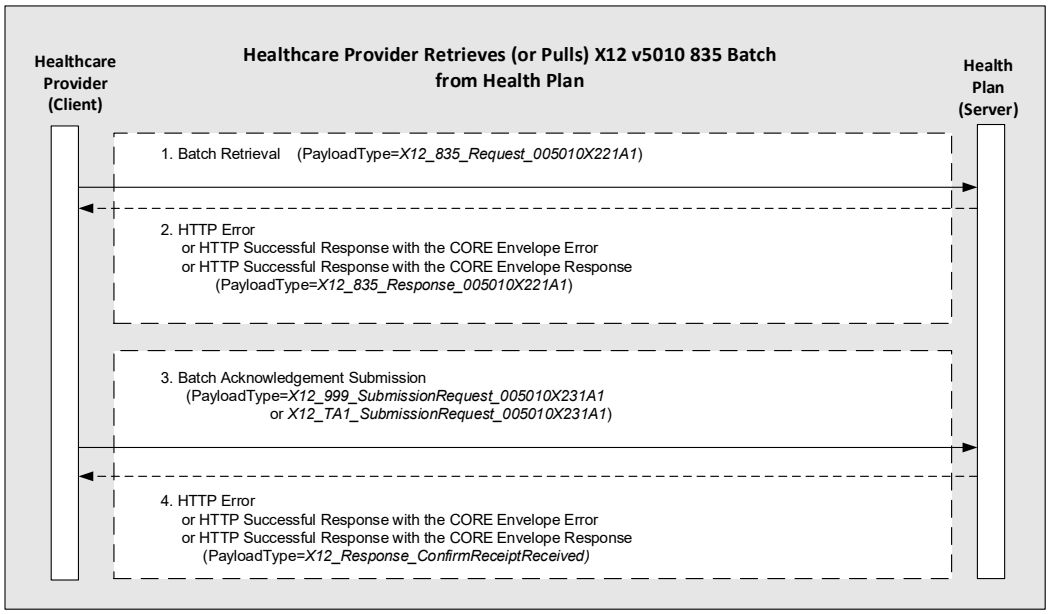
8.3.3.2. Generic Pull

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 10: Health Care Payment/Advice (X12 v5010 835)

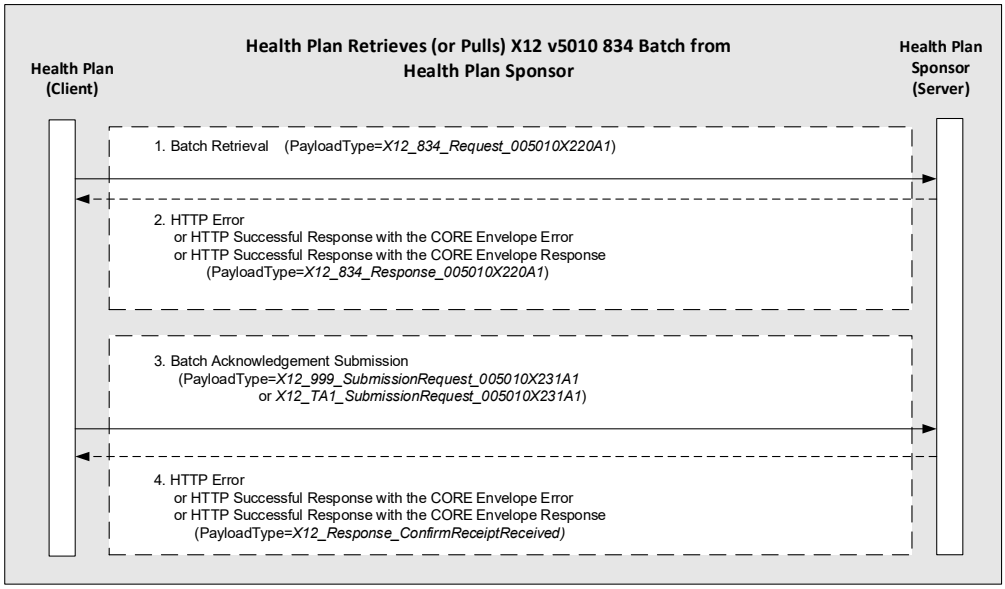


The following describes the *Health Care Payment/Advice* transaction using the *Generic Pull* interaction, as shown in the above diagram.

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
1	A Healthcare Provider(Client) sends a Health Plan (Server) a retrieval request for a Batch of Health Care Payment/Advice requests using PayloadType=*X12_835_Request_005010X221A1*	Benefit Enrollment and Maintenance:
2	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_835_Request_005010X221A1*)	Benefit Enrollment and Maintenance:
3	A Healthcare Provider (Client) submits to a Health Plan (Server) the acknowledgement (PayloadType *X12_999_SubmissionRequest_005010X231A1* or *X12_TA1_SubmissionRequest_005010X231A1*) to the Health Plan.	Implementation Acknowledgement Submission

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
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

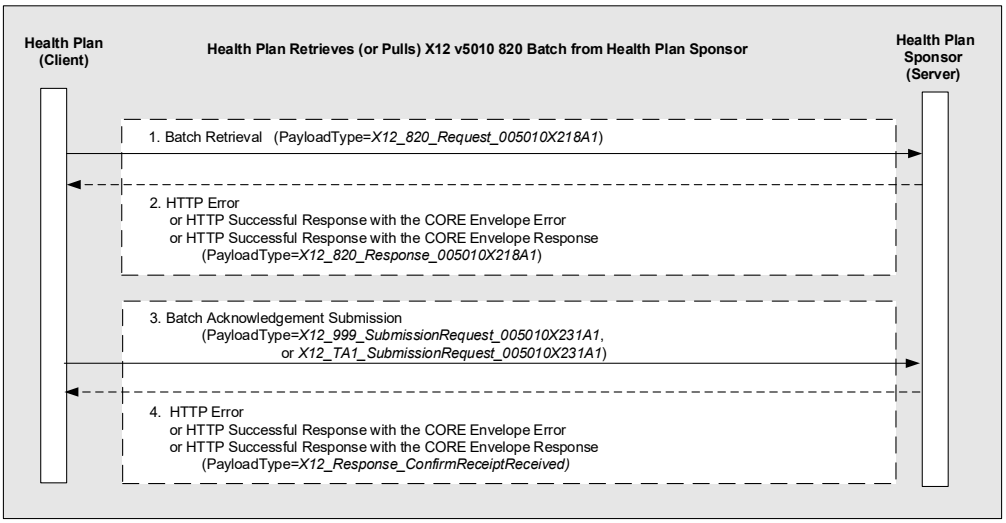
Example 11: Benefit Enrollment and Maintenance (X12 v5010 834)



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:
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 CAQH CORE Connectivity Rule vC1.1.0 and CAQH CORE Connectivity Rule vC2.2.0.	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 12: Payroll Deducted and Other Group Premium Payment for Insurance Products (X12 v5010 820)

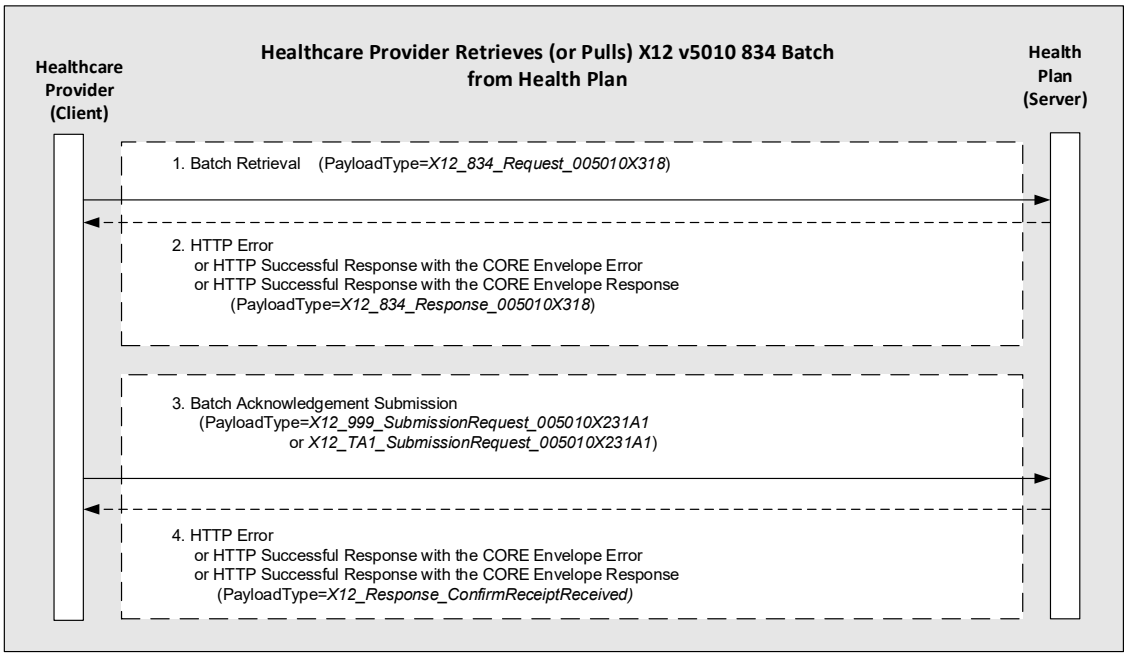


The following describes the *Payroll Deducted and Other Group Premium Payment for Insurance Products* transaction using the *Generic Pull* interaction, as shown in the above diagram.

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 CAQH CORE Connectivity Rule vC1.1.0 and CAQH CORE Connectivity Rule vC2.2.0.	Payroll Deducted and Other Group Premium Payment for Insurance Products (Batch Results Acknowledgment Submission)

Message Sequence	Description	Reference to Payload Type Table Transaction Name Column in the Processing Modes and Payload Types Document
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)

Example 13: Plan Member Reporting (X12 v5010 834)



The following describes the *Plan Member Reporting* 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 Healthcare Provider (Client) sends a Health Plan (Server) a retrieval request for a Batch of Plan Member Reporting requests using PayloadType=*X12_834_Request_005010X318*	Benefit Enrollment and Maintenance
2	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_834_Response_005010X318*)	Benefit Enrollment and Maintenance
3	A Healthcare Provider (Client) submits to a Health Plan (Server) the acknowledgement (PayloadType *X12_999_SubmissionRequest_005010X231A1* or *X12_TA1_SubmissionRequest_005010X231A1*) to the Health Plan.	Implementation Acknowledgement Submission
4	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 (PayloadType=*X12_Response_ConfirmReceiptReceived)*	Implementation Acknowledgement Submission