IBM WebSphere Transformation Extender

Pack for HIPAA EDI

Version 4.2

Note

Before using this information, be sure to read the general information in “Notices” on page 91.

30 June 2006

This edition of this document applies to IBM WebSphere Transformation Extender Pack for HIPAA EDI Version 4.2; and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this document, email DTX_doc_feedback@us.ibm.com. We look forward to hearing from you.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 2006. All rights reserved.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Overview of the Packs for Healthcare

There are three separate IBM Healthcare Packs:

- The IBM WebSphere Transformation Extender Pack for Health Insurance Portability and Accountability Act, Electronic Data Interchange (HIPAA EDI)
- The IBM WebSphere Transformation Extender Pack for Health Level Seven (HL7)
- The IBM WebSphere Transformation Extender Pack for the National Council of Prescription Drug Providers (NCPDP)

The objects (type trees and maps) in the Packs include definitions for the complete ASC X12N standard for the HIPAA, as well as NCPDP, HL7 and CMS (formerly HCFA) formats for NSF, UB-92 and 4010 flat file formats.

The Packs provide healthcare and insurance payer organizations with an infrastructure that:

- Enables compliance with government and industry mandates.
- Controls administrative costs.
- Streamlines business processes.
- Facilitates accuracy and timeliness of information.
- Offers a competitive advantage.
- Conforms to existing systems.
- Adapts to new technologies as they emerge.
- Integrates multiple systems and standards.

Because the Packs are based on core IBM technology, your applications and systems can take advantage of IBM's full range of integration capabilities including:

- Powerful any-to-any and many-to-many data transformation.
- Multi-platform deployment and execution.
- Management of transaction flows between trading partners.
- Importers for interface creation from XML DTDs, copybooks, DBMS catalogs, and other metadata.
- Adapters for commercial applications, messaging middleware, Internet transports, relational databases, file systems, and utilities.

Organization of objects

The Packs are collections of objects, including type trees, sample data, map source files, and compiled image files for utility modules. The contents of the Packs are organized by object type with the intention of facilitating reusability and flexibility.

The Packs for Healthcare objects install in the following directory location:

*install_dir***packs\healthcare_v***n.n**pack*

In this location, *install_dir* indicates the core IBM product installation, *n.n* indicates the current product version, and *pack* indicates the actual Pack for Healthcare that is installed (**hipaa**, **hl7**, or **ncpdp**).

© Copyright IBM Corp. 2006

Included in the Packs for HIPAA EDI; HL7; and NCPDP

The data exchange, transformation, and integration requirements of the healthcare industry range from simple to extremely complex and can vary greatly from one organization to another. The Packs contain predefined objects that give you the functionality and flexibility to develop a wide variety of applications and systems that meet your specific transaction and production requirements.

The Packs contain type trees, maps, sample data, and utility modules. These predefined, industry-specific objects are organized in a way that provides the flexibility to create and deploy a wide variety of applications and systems that address healthcare integration requirements. These objects are constructed to allow consistent behavior and results across all supported platforms and operating systems.

The given type trees define the more commonly used healthcare industry data standards. The given maps, data, and utility modules embrace typical data validation and transformation scenarios.

Type trees

The type trees in these packages support the following healthcare standards:

- CMS, NFS, UB92
- HIPAA
- HL7
- NCPDP

The type trees described here describe the type trees that support the IBM WebSphere Transformation Extender Pack for HIPAA EDI.

See the IBM WebSphere Transformation Extender Pack for HL7 documentation for information about the type trees that support the HL7 standard.

See the IBM WebSphere Transformation Extender Pack for NCPDP documentation for information about the type trees that support the NCPDP standard.

Sample data

Data files are included for the Pack for HIPAA EDI and the Pack for NCPDP (data files are not included in the Pack for HL7). The data files included in the two packs fall into one of two general categories. Either they contain transaction data in one or more of the data exchange formats associated with a particular healthcare industry standard; or they contain application data used by the Compliance Check application for HIPAA X12 data validation.

The sample data files in the Pack for HIPAA EDI and the Pack for NCPDP conform to the following standards:

- HIPAA X12 (included with the Pack for HIPAA EDI)
- HIPAA X12 Compliance Check (included with the Pack for HIPAA EDI)
- CMS (included with the Pack for HIPAA EDI)
- NCPDP (included with the Pack for NCPDP)

Maps

The following types of maps are included in this package:

- HIPAA X12 validation and reporting
- Pass-through maps
- NCPDP reporting
- Data transformation

Utility modules

The utility modules are discussed in ″Compliance Checking″.

Intended use

The Packs are intended for use by a wide variety of healthcare organizations, particularly healthcare providers and insurance payers seeking to fulfill Healthcare Industry Portability and Accountability Act (HIPAA) requirements, and streamline healthcare transaction data integration.

In using the Packs you can send and receive electronic healthcare transactions, receive and transmit data using legacy standards, and integrate with internal applications and data.

Compatibility

The executable objects that you create using this version of the Pack for HIPAA EDI; HL7; or NCPDP can be ported to any Windows, UNIX (AIX, HP, Solaris, and Linux), and OS/390 platform.

General constraints

All healthcare-related type trees developed by IBM are expected to adhere to the current, official implementation guides or published standards documents.

Healthcare data transformation maps are expected to be accurate implementations of published crosswalk or data transformation specifications. However, the examples provided do not take into account your specific requirements. Therefore, all type trees and maps provided are intended for use as examples only.

It is your responsibility as a user to assess suitability and perform appropriate testing before placing any of the Packs for Healthcare objects into a production environment. You will also need to ensure that you have adequately addressed security and privacy considerations in the applications and systems that you develop using the Packs for Healthcare.

File naming conventions

All object filenames in the Packs are lowercase and contain only alphanumeric or underscore (_) characters. For compatibility with the OS/390 batch environment, filenames for executable objects, including executable map names referenced by RUN function calls, do not contain underscore characters in the first 8 characters of the filename and the first 8 characters of executable object names are unique within the scope of the Packs for Healthcare. This restriction does not apply to objects that are not ported directly to the z/OS or UNIX environments such as type trees or map source files.

Chapter 2. Healthcare industry

The documentation provided here discusses technical terms and other information used to exchange data in the Healthcare industry.

Healthcare information exchange participants

Every encounter between a patient and healthcare provider involves the exchange of information. In addition to the basics of patient demographics, symptoms, diagnoses, and treatments, the typical scenario requires the exchange of claims and payment data as well as associated payer, subscriber, eligibility and authorization information. A single encounter can involve the transmission of large volumes of information among several participants.

Healthcare transactions and standards

The exchange of healthcare information can generally be viewed as a transaction between the sender and receiver participants.

Healthcare transactions include (but are not limited to):

- Healthcare claim or encounter.
- Claim payment and remittance advice.
- Healthcare claim status.
- Coordination of benefits.
- Eligibility for a health plan.
- Referral certification and authorization.
- Enrollment and un-enrollment in a health plan.
- Premium payments.

These transactions may be transmitted electronically in compliance with healthcare transaction standards.

Healthcare data exchange standards allow the accurate and timely exchange of information between healthcare organizations. For example, a simple benefits inquiry can take 20 minutes on the phone. Using electronic data interchange (EDI); this type of request can be processed almost immediately, without the need for a call to the insurer's customer service center.

Healthcare electronic data interchange initiatives

The entire healthcare industry is facing increased pressure to implement productivity and quality improvements while reducing costs. The use of electronic data interchange and industry-specific data exchange standards for healthcare transaction data is a potential source of significant benefits in these areas.

HIPAA legislation

In 1996, legislation was passed to improve the overall healthcare administrative system. This legislation is known as the Health Insurance Portability and Accountability Act (HIPAA).

© Copyright IBM Corp. 2006

The Pack for HIPAA EDI specifically addresses the administrative simplification aspects of HIPAA legislation - the standardization of electronic patient health, administrative, and financial data.

HIPAA regulations affect payers, health plans, clearinghouses, and those providers who conduct financial and administrative transactions electronically.

Healthcare standards organizations

There are a number of organizations that participate in the development and publication of healthcare data standards. Because of their importance in the area of healthcare data exchange standards and formats, IBM focuses on the standards and formats developed and maintained by the following four organizations:

- ANSI ASC X12N
- NCPDP
- HL7
- CMS

ANSI ASC X12N

The Accredited Standards Committee (ASC) of the American National Standards Institute (ANSI) maintains the X12 standard. X12 is the dominant EDI standard in North America. The transaction sets included in the X12 standard cover a wide range of industries and business functions - including the exchange of healthcare data. The Healthcare Task Group of the Insurance Subcommittee (also known as X12N) is the designated standards maintenance organization (DSMO) for most of the finalized HIPAA transaction set standards.

This version of the Pack for HIPAA EDI includes type tree definitions for all of the finalized HIPAA X12 transaction sets and the HIPAA X12 Addenda.

Also included are maps and applications for transforming and validating HIPAA X12 transaction data.

NCPDP

The National Council for Prescription Drug Programs (NCPDP) is another HIPAA-designated standards maintenance organization (DSMO) for finalized HIPAA transactions. NCPDP standards are used exclusively in the retail pharmacy sector.

The Pack for NCPDP includes type tree definitions for all of the finalized HIPAA NCPDP Telecommunications V5.1 transaction sets and Batch V1.0 format. The package also contains a type tree with NCPDP V3.2 transaction sets and Batch V1.1 format.

HL7

Health Level Seven (HL7) is an ANSI-accredited standards organization operating in the healthcare arena. HL7's domain is clinical and administrative data.

The HL7 produces standards for the exchange, management, and integration of data, intended to promote interoperability between healthcare information systems.

The Pack for HL7 includes type tree definitions for Version 2.1, 2.2, 2.3 and 2.5 of the HL7 standards.

CMS

The Center for Medicare and Medicaid Services (CMS), formerly known as ″HCFA″, is the supervisory organization for all public expenditures for healthcare. CMS provides health insurance for over 74 million Americans through Medicare, Medicaid, and the State Child Health Insurance Program. CMS is not a standards organization but their published interface formats are used extensively throughout the United States.

The Pack for HIPAA EDI includes type tree definitions for the following formats:

- NSF Claims and Coordination of Benefits - Version 3.01
- UB-92 Claims and Coordination of Benefits - Versions 5.0 and 6.0
- 4010 Flat Files for Professional and Institutional Claims and Coordination of Benefits (837), Payments (835) and Claim Status Request/Response (276/277)

Note: CMS was formerly the Healthcare Financing Administration (HCFA). Because many CMS documents, standards, and Web sites still refer to HCFA, both CMS and HCFA are used in this guide and in IBM Transformation Extender Packs for Healthcare.

Other healthcare data exchange organizations

Another key participant in the development and interpretation of Healthcare data exchange standards, HIPAA transaction standards in particular, is the Workgroup for Electronic Data Interchange (WEDI) Strategic National Implementation Process (SNIP) Transactions Work Group. WEDI/SNIP has documented an approach for classifying the various HIPAA transaction compliance requirements into types (formerly referred to as ″levels″) of testing. This approach has gained widespread acceptance in the Healthcare industry. The Compliance Check application and the HIPAA X12 type tree definitions in the Pack for HIPAA EDI adhere to WEDI/SNIP guidelines.

Chapter 3. HIPAA EDI Type trees

The documentation provided here discusses the type trees included with the Pack for HIPAA EDI.

The type trees are organized under the following headings:
- HIPAA X12 Type Trees
- Utility Type Trees
- CMS Type Trees

Overview

The Pack for HIPAA EDI includes type trees with data definitions for significant healthcare industry standards. The type trees in these packages depict the structure, relationships, and data element attributes as described in published specification documents. In practical terms, this means data that meets the specification requirements is considered valid, while data that does not meet the requirements is rejected. Other type tree attributes include the following:
- Compatibility with other IBM product offerings, such as Trading Manager.
- Flexibility in modifying or merging with other type trees.
- Ability to support reliable and efficient mapping.
- Easily maintainable

There are two kinds of type trees in the packages: the trees that define the data exchange formats associated with Healthcare industry data standards and the trees that describe data formats used by application maps in the packages.

The type trees for healthcare application maps are described in ″Compliance Checking″ documentation.

What the type trees support

The set of type trees included with the Pack for HIPAA EDI support WEDI/SNIP Type 1, Type 2, Type 3, Type 4, and HIPAA finalized (round 2) Addenda.

The type tree variants that enforce the HIPAA Type 2, Type 3, and Type 4 tests are designed to work interchangeably (that is, ″plug-and-play″) in data transformation maps. Wherever possible, HIPAA X12 transaction set definitions and their corresponding addenda definitions are consistent with the intent to preserve the ability to merge and modify HIPAA X12 transaction set definitions and allow maximum flexibility.

The type trees do not produce acknowledgment responses. In order to generate acknowledgment responses, the Compliance Check application map must be used.

About HIPAA X12 finalized addenda

The type trees in this package include HIPAA finalized Addenda transaction sets. You can quickly validate your HIPAA X12 transmission data in accordance with the WEDI/SNIP types of testing by using a corresponding type tree.

Compliance checking and reporting for HIPAA X12 Addenda transaction sets is consistent with the compliance checking and reporting for the original transaction set definitions.

General type tree information

Most of the type trees in the Pack for HIPAA EDI, the Pack for HL7, and the Pack for NCPDP include a category: **%_Type_Tree_Information**. Double-click on the **%_Description or %_Revision_History** items to see a narrative description of the type tree or the type tree revision history.

HIPAA X12 type trees

HIPAA X12 type trees describe the data exchange formats associated with HIPAA X12 transaction sets.

The following type trees are included with the Pack for HIPAA EDI:
- hipaa_x12_4010.mtt
- hipaa_x12_4010_type_1.mtt
- hipaa_x12_4010(a1)_type_2.mtt
- hipaa_x12_4010(a1)_type_4.mtt

The definitions of the HIPAA X12 type trees in the Pack for HIPAA EDI are based on the following criteria:
- X12 standards documents and table data published by DISA.
- HIPAA X12 implementation guide specification documents published by Washington Publishing Company.

For the purpose of the Pack for HIPAA EDI, published documents take precedence in cases where the table data and the standards documents do not agree.

The origin of the HIPAA X12 type trees in this package is from the X12 standard type tree definitions in the Pack for HIPAA EDI. HIPAA-specific modifications were made to the X12 standard definitions in accordance with the HIPAA X12 Implementation Guides.

The HIPAA X12 type trees are described in the following sections and are listed by filename.

hipaa_x12_4010.mtt

The **hipaa_x12_4010.mtt** type tree describes the HIPAA X12 transaction sets and enveloping structures associated with the following finalized HIPAA X12 transaction sets:

Transaction	Description	Version
270	Health Plan Eligibility Inquiry	004010X092/a1
271	Health Plan Eligibility Response	004010X092/a1
276	Claim Status Request	004010X093/a1
277	Claim Status Response	004010X093/a1
278	Referral Request and Response	004010X094/a1
820	Health Plan Premium Payment	004010X061/a1

Transaction	Description	Version
834	Health Plan Enrollment	004010X095/a1
835	Health Care Claim Payment	004010X091/a1
837(I)	Health Care Claim and COB: Institutional	004010X096/a1
837(D)	Health Care Claim and COB: Dental	004010X097/a1
837(P)	Health Care Claim and COB: Professional	004010X098/a1
275	Additional Information to support a health care claim or Encounter	004050X151
277	Claim Request for Additional Information	004050X150

This type tree enforces compliance of HIPAA X12 transmission data with the following WEDI/SNIP transaction compliance types:
- Type 1 - EDI Standard Integrity Testing
- Type 2 - HIPAA Implementation Guide Requirement Testing rules
- Type 3 - HIPAA balance testing
- Type 4 - HIPAA inter-segment situation testing

For a detailed description of Type 1, 2, 3 and 4 testing, see WEDI/SNIP HIPAA Transaction Compliance Types.

Partner X12 Inbound and **Outbound Transmission EDI** are the top-level definitions in this type tree.

The **hipaa_x12_4010.mtt** type tree is utilized in the following maps.

Addenda map names are indicated in parenthesis in the following list. For example, the second map name in the list: **hipaa_276i(_4010a1)_to_cms_flat.mms** represents two separate maps: **hipaa_276i_to_cms_flat.mms** and **hipaa_276i_4010a1_to_cms_flat.mms**.
- **hipaa_x12_4010_pass_through.mms**
- **hipaa_276i(_4010a1)_to_cms_flat.mms**
- **hipaa_276p(_4010a1)_to_cms_flat.mms**
- **cms_277i(_4010a1)_flat_to_hipaa.mms**
- **cms_277p(_4010a1)_flat_to_hipaa.mms**
- **hipaa_835i(_4010a1)_to_cms_flat.mms**
- **cms_835i(_4010a1)_flat_to_hipaa.mms**
- **hipaa_835p(_4010a1)_to_cms_flat.mms**
- **cms_835p(_4010a1)_flat_to_hipaa.mms**
- **hipaa_837i(_4010a1)_to_cms_flat.mms**
- **cms_837i(_4010a1)_flat_to_hipaa.mms**
- **hipaa_837i_to_ub92_v6_0.mms**
- **ub92_v6_0_to_hipaa_837i.mms**
- **hipaa_837p(_4010a1)_to_cms_flat.mms**
- **cms_837p(_4010a1)_flat_to_hipaa.mms**
- **hipaa_837p_to_nsf_v3_01.mms**
- **nsf_claim_v3_01_to_hipaa_837p.mms**

hipaa_x12_4010_type_1.mtt

The **hipaa_x12_4010_type_1.mtt** type tree describes the HIPAA X12 transaction sets and enveloping structures associated with the finalized HIPAA X12 transaction standards.

This type tree enforces compliance of HIPAA X12 transmission data with WEDI/SNIP Type 1-EDI Standard Integrity Testing rules, including checks for:
- valid segments
- segment order
- maximum segment and loop occurrences
- element attributes (for example, size and data type)
- adherence to X12 syntax rules as defined in the X12 Standard.

Partner X12 Inbound and **Outbound Transmission EDI** are the top-level definitions in this type tree.

In order to prevent data from being rejected by the data transformation map, the **hipaa_x12_4010_type_1.mtt** type tree should be used as the input type tree whenever the HIPAA X12 data has been validated against only the Type 1 - EDI Standard Integrity Testing compliance checks.

hipaa_x12_4010(a1)_type_2.mtt

The **hipaa_x12_4010_type_2.mtt** type tree describes the HIPAA X12 transaction sets and enveloping structures associated with the finalized HIPAA X12 transaction standards.

The **hipaa_x12_4010a1_type_2.mtt** contains addenda HIPAA X12 transaction standards. **This type tree enforces compliance of HIPAA X12 transmission data for the following:**
- Type 1 Standard Integrity Testing
- HIPAA Implementation Guide Requirement Testing.

Partner X12 Inbound and **Outbound Transmission EDI** are the top-level definitions in this type tree.

The hipaa_x12_4010(a1)_type_2.mtt; and **hipaa_x12_4010.mtt** type trees can be used interchangeably for mapping because they have the same structure and elements. The only differences between them are the additional component rules in **hipaa_x12_4010.mtt** that enforce the balance checking and the inter-segment situational checks.

In order to prevent data from being rejected by the data transformation map, the **hipaa_x12_4010(a1)_type_2.mtt** type tree should be used as the input type tree whenever the HIPAA X12 data has been validated against the Type 2 - HIPAA Implementation Guide Requirement Testing compliance checks.

The **hipaa_x12_4010(a1)_type_2.mtt** type tree may be used in place of **hipaa_x12_4010.mtt** in the following maps:
- hipaa_276i(_4010a1)_to_cms_flat.mms
- hipaa_276p(_4010a1)_to_cms_flat.mms
- cms_277i(_4010a1)_flat_to_hipaa.mms
- cms_277p(_4010a1)_flat_to_hipaa.mms

- hipaa_835i(_4010a1)_to_cms_flat.mms
- cms_835i(_4010a1)_flat_to_hipaa.mms
- hipaa_835p(_4010a1)_to_cms_flat.mms
- cms_835p(_4010a1)_flat_to_hipaa.mms
- hipaa_837i(_4010a1)_to_cms_flat.mms
- cms_837i(_4010a1)_flat_to_hipaa.mms
- hipaa_837i_to_ub92_v6_0.mms
- ub92_v6_0_to_hipaa_837i.mms
- hipaa_837p(_4010a1)_to_cms_flat.mms
- cms_837p(_4010a1)_flat_to_hipaa.mms
- hipaa_837p_to_nsf_v3_01.mms
- nsf_claim_v3_01_to_hipaa_837p.mms

hipaa_x12_4010(a1)_type_4.mtt

The **hipaa_x12_4010_type_4.mtt** type tree describes HIPAA the X12 transaction sets and enveloping structures associated with the finalized HIPAA X12 transaction standards, while the **hipaa_x12_4010a1_type_4.mtt** contains addenda HIPAA X12 transaction standards.

This type tree provides HIPAA Inter-Segment Situation testing. See ″Type 4 - HIPAA Inter-Segment Situation Testing″.

Partner X12 Inbound and **Outbound Transmission EDI** are the top-level definitions in this type tree.

Employer Identification Number (EIN) support

Starting in June 2004, Code 65 for EIN is mandated for large entities. Small entities are not required to be compliant until August 2005. For this reason, the support for EIN Code 65 is included as an additional valid code for EIN until the mandate is final.

Implementation guide changes

Below is a list of the changes in the Implementation Guides that are included to support the mandate code.

Change 1

V4010X061 - V4010X061A1 -820 Transaction Set (Implementation Guide page 63, Loop 1000B, N103)

Code ′65′ - National employer Identification. This is required for a HIPAA compliant implementation when the National employer ID is mandated. Since large entities are now mandated to use this code , until it becomes mandatory for all, both codes ″FI″ and ″65″ are accepted as HIPAA compliant identifiers. Code Code ′65′ is not a valid X12 value for N103. For this reason, when using this code, be aware that it will fail Type 1 validation.

Change 2

V4010X095 - V4010X095A1 - 834 Transaction Set (Implementation Guide page 36, Loop 1000A, N103)

Code ′ZZ′ - Mutually Defined - The value ′ZZ′ when used in this data element is the ″HIPAA Employer Identifier″ once this identifier has been adopted. The Secretary of the Department of Health and Human Services has adopted Code ′65′ as a Standard Employer Identifier to use in this transactions when mandated. Since large entities are now mandated to use code ″65″ for the Employer Identifier, until it becomes mandatory for all, codes ″FI″, ″ZZ″ and ″65″ are accepted as HIPAA compliant identifiers.

Note: Code ′65′ is not a valid X12 value for N103. For this reason, when using this code, be aware that it will fail Type 1 validation.

Change 3

V4010X095 - V4010X095A1 - 834 Transaction Set (Implementation Guide page 91, Loop 2100D, NM108)

Code ′ZZ′- Mutually defined - This code can be used in this NM108 for the National Employer Identifier where a standard code is not yet mandated. The Secretary of the Department of Health and Human Services has adopted Code ′65′ as a Standard Employer Identifier to use in this transitions when mandated. Since large entities are now mandated to use code ″65″ for the Employer Identifier, until it becomes mandatory for all, codes both ″ZZ″ and ″65″ are accepted as HIPAA compliant identifiers.

Type tree modifications

In order to use the new identifier code, the following tree modifications were made. Each tree modification is numbered according to the number of the change (Change 1, 2, and 3) described in the preceding paragraphs.

These rules may be changed to customize your needs, for example to strictly adhere to the mandated EIN code, remove the other codes from the list.

Change 1: hipaa_x12_4010_segment.mtt

Loop_1000B_N1 Segment V4010X061 ANSI EDI
Loop_1000B_N1 Segment V4010X061A1 ANSI EDI

Rule on Component 5 of both groups:

```
WHEN(PRESENT($),ONERROR(MEMBER($,{″FI″,″65″}),
″E*3*E*C*7*C*D*″+$+″*D″) )
```

Change 1: hipaa_x12_4010_segment.mtt

Unknown_N1 Segment V4010X061 ANSI EDI
Unknown_N1 Segment V4010X061A1 ANSI EDI

Rule on Component 5 of both groups:

```
WHEN(PRESENT($),
ONERROR(MEMBER($,{″FI″,″65″}),
″E*3*E*C*7*C*D*″+$+″*D″) )
```

Change 2: hipaa_x12_4010.mtt hipaa_x12_4010_type_2.mtt hipaa_x12_4010a1_type_2.mtt hipaa_x12_4010_type_4.mtt hipaa_x12_4010a1_type_4.mtt

N1_PR_Payer N1 Segment V4010X061 ANSI EDI
N1_PR_Payer N1 Segment V4010X061A1 ANSI EDI

Rule on Component 3 of both groups:

MEMBER(IDCdQual’r Element:$,{″FI″,″65″})

Change 2: hipaa_x12_4010_segment.mtt

Loop_1000A_N1 Segment V4010X095 ANSI EDI
Loop_1000A_N1 Segment V4010X095A1 ANSI EDI

Rule on component 5:

WHEN(PRESENT($),ONERROR(MEMBER($,{″FI″,″ZZ″,″65″}),
″E*3*E*C*7*C*D*″+$+″*D″))

Change 2: hipaa_x12_4010_segment.mtt

Unknown_N1 Segment V4010X095 ANSI EDI
Unknown_N1 Segment V4010X095A1 ANSI EDI

Rule on component 5:

WHEN(PRESENT($),ONERROR(MEMBER($,
{″94″,″FI″,″NI″,″XV″″ZZ″,″65″}),
″E*3*E*C*7*C*D*″+$+″*D″))

Change 2: hipaa_x12_4010.mtt hipaa_x12_4010_type_2.mtt hipaa_x12_4010a1_type_2.mtt hipaa_x12_4010_type_4.mtt hipaa_x12_4010a1_type_4.mtt

Header_Level Names Unordered_Group #834 Inbound Partner Set V4010X095 ANSI EDI

Header_Level Names Unordered_Group #834 Outbound Partner Set V4010X095 ANSI EDI

Header_Level Names Unordered_Group #834 Inbound Partner Set V4010X095A1 ANSI EDI

Header_Level Names Unordered_Group #834 Outbound Partner Set V4010X095A1 ANSI EDI

Rule on component 1:

MEMBER(IDCdQual’r Element:IDCd Element MComposite:$,
{″FI″, /* Federal Taxpayer’s Identification Number */
″ZZ″, /* Mutually Defined - HIPAA Employer Identifier */
″65″})/*EIN */

Change 3: hipaa_x12_4010_segment.mtt

Loop_2100D_NM1 Segment V4010X095 ANSI EDI
Loop_2100D_NM1 Segment V4010X095A1 ANSI EDI

Rule on component 10:

WHEN(PRESENT($),
ONERROR(MEMBER($,{″ZZ″,″65″}),
″E*8*E*C*7*C*D*″ + $ + ″*D″)))

Change 3: hipaa_x12_4010_segment.mtt

UNKNOWN_NM1 Segment V4010X095 ANSI EDI
UNKNOWN_NM1 Segment V4010X095A1 ANSI EDI

Rule on component 10:

WHEN(PRESENT($),
ONERROR(MEMBER($,
{″34″,″FI″,″XX″,″ZZ″,″65″}),
″E*8*E*C*7*C*D*″ + $ + ″*D″)))

Change 3: hipaa_x12_4010.mtt hipaa_x12_4010_type_2.mtt hipaa_x12_4010a1_type_2.mtt hipaa_x12_4010_type_4.mtt hipaa_x12_4010a1_type_4.mtt

Member_Employer NM1 Loop #834 Inbound Partner Set V4010X095A1 ANSI EDI

Member_Employer NM1 Loop #834 Outbound Partner Set V4010X095A1 ANSI EDI

Member_Employer NM1 Loop #834 Inbound Partner Set V4010X095 ANSI EDI

Member_Employer NM1 Loop #834 Outbound Partner Set V4010X095 ANSI EDI

Rule on Component 1 of all groups:

WHEN(EntityTypeQual’r Element:$ = ″1″ & PRESENT(NameLastOrgName Element:$),
PRESENT(NameFirst Element:$)) &
ABSENT(NamePrefix Element:$) &
WHEN(PRESENT(IDCd Element MComposite:$),

MEMBER(IDCdQual’r Element:IDCd Element MComposite:$,
{″ZZ″,″65″})) &
ABSENT(EntityRelationshipCd Element:$)

CMS type trees

The CMS type trees describe the healthcare data interface formats as specified by the Center for Medicare and Medicaid Services (CMS). The following type trees are delivered with the Pack for HIPAA EDI:

- cms_276i_277i(_4010a1)_flat.mtt
- cms_276p_277p(_4010a1)_flat.mtt
- cms_835i(_4010a1)_flat.mt

- cms_835p(_4010a1)_flat.mtt
- cms_837i(_4010a1)_flat.mtt
- cms_837p(_4010a1)_flat.mtt
- nsf_claim_v3_01.mtt
- ub92_v5_0.mtt
- ub92_v6_0.mtt

cms_276i_277i(_4010a1)_flat.mtt

The **cms_276i_277i_4010a1_flat.mtt** type tree defines the contents of the CMS (formerly HCFA) Part A 276 and 277 Institutional Health Care Claim Status Request and Response Flat files that correspond with the HIPAA 4010X093 Addenda formats.

The **cms_276i_277i_flat.mtt** type tree contains a definition of the CMS Part A Healthcare Claim Status Request (276) and Response (277) flat file formats.

The top-level groups in this type tree are **cms_276i_flat file data** and **cms_277i_flat file data**.

The **cms_276i_277i_flat.mtt** type tree is utilized in the following maps:

- cms_276i(_4010a1)_flat_pass_through.mms
- cms_277i(_4010a1)_flat_pass_through.mms
- hipaa_267i(_4010a1)_to_cms_flat.mms
- cms277i(_4010a1)_flat_to_hipaa.mms

cms_276p_277p(_4010a1)_flat.mtt

The **cms_276p_277p_4010a1_flat.mtt** contains a definition of the CMS Part B Healthcare Claim Status Request and Response flat file format. The CMS_276P_Flat File Data object defines the CMS Part B Healthcare Claim Status Request flat file and the CMS_277P_Flat File Data object defines the CMS Part B Healthcare Claim Status Response flat file.

The **cms_276p_277p_flat.mtt** type tree contains a definition of the CMS Part B Healthcare Claim Status Request (276) and Response (277) flat file formats.

The top-level groups in this type tree are **cms_276p_flat file data** and **cms_277p_flat file data**.

The **cms_276p_277p_flat.mtt** type tree is utilized in the following maps:

- cms_276p_flat_pass_through.mms
- cms_277p_flat_pass_through.mms
- hipaa_267p_to_cms_flat.mms
- cms_276p_277p_4010a1.mtt

cms_835i(_4010a1)_flat.mtt

The **cms_835i_4010a1_flat.mtt** type tree defines the contents of the CMS (formerly HCFA) Part A 835 Institutional Health Care Claim Payment Flat file that corresponds with the HIPAA 4010X091 Addenda format.

The **cms_835i_flat.mtt** type tree contains the definition of the CMS Part A Electronic Remittance Advice (835) flat file format.

The top-level group in this type tree is **CMS_835i_Flat**.

The **cms_835i_flat.mtt** type tree is utilized in the following maps:
- cms_835i_flat_pass_through.mms
- cms_835i_flat_to_hipaa.mms

cms_835p(_4010a1)_flat.mtt

The **cms_835p_4010a1_flat.mtt** type tree defines the contents of the CMS (formerly HCFA) Part B 835 Professional Health Care Claim Payment Flat file that corresponds with the HIPAA 4010X091 Addenda format.

Developed in accordance with CMS specifications, the **cms_835p_flat.mtt** type tree contains the definition of the CMS Part B Electronic Remittance Advice (835) flat file format.

The top-level group in this type tree is **CMS_835P_flat file data**.

The **cms_835p_flat.mtt** type tree is utilized in the following maps:
- cms_835p_flat_pass_through.mms
- cms_835p_flat_to_hipaa.mms
- hipaa_835p_to_cms_flat.mms
- cms_837i_4010a1_flat.mtt

cms_837i(_4010a1)_flat.mtt

The **cms_837i_4010a1_flat.mtt** type tree defines the contents of the CMS (formerly HCFA) Part A 837 Institutional Claim Flat file that corresponds with the HIPAA 4010X096 Addenda format.

The **cms_837i_flat.mtt** type tree contains the definition of the CMS Part A Institutional Claim and Coordination of Benefits (837) flat file format.

cms_837i_flat file data is the top-level group in this type tree.

The **cms_837i_flat.mtt** type tree is utilized in the following maps:
- cms_837i_flat_pass_through.mms
- cms_837i_flat_to_hipaa.mms
- hipaa_837i_to_cms_flat.mms

cms_837p(_4010a1)_flat.mtt

The **cms_837p_4010a1_flat.mtt** type tree defines the contents of the CMS (formerly HCFA) Part B 837 Professional Claim Flat file that corresponds with the HIPAA 4010X096 Addenda format.

Developed in accordance with CMS specifications, the **cms_837p_flat.mtt** type tree contains the definition of the CMS Part B Professional Claim and Coordination of Benefits (837) flat file format.

cms_837p_flat file data is the top-level group in this type tree.

The **cms_837p_flat.mtt** type tree is utilized in the following maps:
- cms_837p_flat_pass_through.mms
- cms_837p_flat_to_hipaa.mms

- hipaa_837p_to_cms_flat.mms

nsf_claim_v3_01.mtt

The **nfs_claim_v3_01.mtt** type tree contains a definition of the CMS (formerly HCFA) Part B National Standards Format (NSF) Version 3.01 format for Claims and Coordination of Benefits.

The top-level group in this type tree is **FileGrp structure NSF_Claim**.

The **nsf_claim_v3_01.mtt** type tree is utilized in the following maps:
- nsf_claim_v3_01_flat_pass_through.mms
- nsf_claim_v3_01_to_hipaa_837p.mms
- hipaa_837p_to_nsf_v3_01.mms

ub92_v5_0.mtt

The **ub92_v5_0.mtt** type tree contains a definition of the CMS (formerly HCFA) Part A UB-92 Version 5.0 format for Healthcare Claims.

The top-level group in this type tree is **File structure ub92claim**.

ub92_v6_0.mtt

The **ub92_v6_0.mtt** type tree contains a definition of the CMS (formerly HCFA) Part A UB-92 Version 6.0 format for Claims and Coordination of Benefits.

The top-level group in this type tree is **File structure UB92**.

The **ub92_v6_0.mtt** type tree is utilized in the following maps:
- ub92_v6_0_pass_through.mms
- ub92_v6_0_to_hipaa_837i.mms
- hipaa_837i_to_ub92_v6_0.mms

Utility type trees

The **hippa_x12_4010_ruleless.mtt** utility type tree is included with the Pack for HIPAA EDI:

hippa_x12_4010_ruleless.mtt

The **hipaa_x12_4010_ruleless.mtt** type tree contains the minimum number of component rules and restrictions required to properly identify the structure of an X12 transmission without doing complex validation on input.

This type tree can be used for transformation when the data is known to be valid after it has been checked for compliance. Performance is therefore enhanced as the data does not need to re-checked for validity.

Chapter 4. Maps

The Pack for HIPAA EDI includes the following types of maps:

- **data transformation maps** - to transform data from one healthcare data format to another
- **pass-through maps** - to validate data definitions against healthcare care industry standards
- **compliance checking maps** - to verify compliance with HIPAA standards. See the Compliance Checking documentation for a discussion of these maps

Data transformation maps

The data transformation maps included in the Pack for HIPAA EDI convert data from one healthcare data format to another.

All data transformation maps have self-describing names with **_to_** separating the names of the source and target data formats. The executable map name is the same as the map source file name. By default, an execution summary audit log named <*executable-map-name*>**.log** will be created in the map directory when the data transformation map is executed.

The following is a list of data transformation maps included with the Packs for HIPAA EDI that fall into four categories:

- Institutional Claims and Coordination of Benefits
- Professional Claims and Coordination of Benefits
- Healthcare Claims Payments
- Healthcare Claims Status Request and Response

After installing the Pack for HIPAA EDI, the maps are located in the

*install_dir***packs****healthcare_v***n.n***hipaa****maps** subdirectory.

Institutional claims and coordination of benefits

The following maps are used for institutional claims and benefit coordination:

- cms_837i (_4010a1)_flat_to_hipaa.mms
- hipaa_837i(_4010a1)_to_cms_flat.mms
- hipaa_837i_to_ub92_v6_0.mms
- ub92_v6_0_to_hipaa_837i.mms

In the following map descriptions, the portion of the file name in parenthesis indicates a map associated with the Addenda. For example, in the first map description two actual maps exist: **cms_837i_flat_to_hipaa.mms** and **cms_837i_4010a1_flat_to_hipaa.mms**. In this case, the second map is Addenda related.

cms_837i (_4010a1)_flat_to_hipaa.mms

This map accepts CMS (formerly HCFA) Part A 4010 claims in flat-file format and generates a HIPAA transmission containing institutional claim and coordination of benefits (837I) transaction sets.

© Copyright IBM Corp. 2006

hipaa_837i(_4010a1)_to_cms_flat.mms

This map accepts HIPAA institutional claim (837I) data and generates the CMS (formerly HCFA) Part A flat file.

hipaa_837i_to_ub92_v6_0.mms

This map accepts HIPAA institutional claim (837I) data and generates a UB-92 Version 6.0 claim submission in flat-file format.

The **hipaa_837i_to_ub92_v6_0.mms** map batches claims by bill type. Each new provider/bill type combination results in a new 10-95 loop. A pre-processing step sorts and groups the service line details that eventually populate the 50, 60, and 61 output records.

ub92_v6_0_to_hipaa_837i.mms

This map accepts a UB-92 Version 6.0 claim submission and generates a HIPAA transmission containing institutional claim and coordination of benefits (837I) transaction sets.

Professional claims and coordination of benefits

This section describes the following maps:

-
- cms_837p(_4010a1)_flat_to_hipaa.mms
- hipaa_837p(_4010a1)_to_cms_flat.mms
- hipaa_837p_to_nsf_v3_01.mms
- nsf_claim_v3_01_to_hipaa_837p.mms

cms_837p(_4010a1)_flat_to_hipaa.mms

This map accepts CMS (formerly HCFA) Part B 4010 claim data in flat-file format and generates a HIPAA transmission containing professional claim and coordination of benefits (837P) transaction sets.

hipaa_837p(_4010a1)_to_cms_flat.mms

This map accepts HIPAA professional claim and coordination of benefits (837P) data and generates the CMS (formerly HCFA) Part B flat file.

hipaa_837p_to_nsf_v3_01.mms

This map accepts HIPAA professional claim and coordination of benefits (837P) data and generates an NSF Version 3.01 claim submission file.

nsf_claim_v3_01_to_hipaa_837p.mms

This map accepts an NFS Version 3.01 claim submission file and generates a HIPAA X12 transmission containing professional claim and coordination of benefits (837P) transaction sets. This map does not convert the claims and service data.

Healthcare claims payments

This section describes the following maps:

- cms_835i(_4010a1)_flat_to_hipaa.mms
- cms_835p(_4010a1)_flat_to_hipaa.mms
- hipaa_835i(_4010a1)_to_cms_flat.mms
- hipaa_835p(_4010a1)_to_cms_flat.mms

cms_835i(_4010a1)_flat_to_hipaa.mms

This map is for data transformation from the CMS 835 Institutional flat file to a HIPAA transmission containing 835 (payment) transactions.

cms_835p(_4010a1)_flat_to_hipaa.mms

This map is for data transformation from the CMS 835 professional flat file format to HIPAA transmissions containing 835 (payment) transactions.

hipaa_835i(_4010a1)_to_cms_flat.mms

This map accepts HIPAA institutional claim data and generates the CMS 835I flat file.

hipaa_835p(_4010a1)_to_cms_flat.mms

This map is for HIPAA transmissions containing 835 (payment) transactions to the CMS 835 professional flat-file format.

Healthcare claims status request and response

This section describes the following maps:

- hipaa_276i(_4010a1)_to_cms_flat.mms
- cms_277i(_4010a1)_flat_to_hipaa.mms
- hipaa_276p(_4010a1)_to_cms_flat.mms
- cms_277p(_4010a1)_flat_to_hipaa.mms

hipaa_276i(_4010a1)_to_cms_flat.mms

This map accepts a HIPAA X12 transmission containing institutional claim status request (276I) transaction sets and generates institutional claim status requests in CMS Part A flat file format.

cms_277i(_4010a1)_flat_to_hipaa.mms

This map accepts claim status responses in CMS Part A flat file format and generates a HIPAA X12 transmission containing institutional claim status response (277I) transaction sets.

hipaa_276p(_4010a1)_to_cms_flat.mms

This map accepts a HIPAA X12 transmission containing professional claim status request (276P) transaction sets and generates professional claim status requests in CMS Part B flat file format.

cms_277p(_4010a1)_flat_to_hipaa.mms

This map accepts claim status responses in CMS Part B flat file format and generates a HIPAA X12 transmission containing professional claim status response (277P) transaction sets.

Maps for data validation

The data validation maps included in the Pack for HIPAA EDI can be used for data validation and reporting. These maps are either pass-through maps, or they are related to the Compliance Check application for HIPAA X12 data validation.

Pass-through maps

Also known as validation maps, pass-through maps validate data definitions by using the same type tree for the input cards as the output cards.

Pass-through maps have self-describing names with **_pass_through** following the name of the data format. The executable map name is the same as the map source file name. By default, an execution summary audit log named <*executable-map-name*>**.log** will be created in the map directory when the pass-through map is executed.

This section describes the following maps:
- hipaa_x12_4010_pass_through.mms
- cms_276i(_4010a1)_flat_pass_through.mms
- cms_276p(_4010a1)_flat_pass_through.mms
- cms_277i(_4010a1)_flat_pass_through.mms
- cms_277p(_4010a1)_flat_pass_through.mms
- cms_835i(_4010a1)_flat_pass_through.mms
- cms_835p(_4010a1)_flat_pass_through.mms
- cms_837i(_4010a1)_flat_pass_through.mms
- cms_837p(_4010a1)_flat_pass_through.mms
- nsf_claim_v3_01_pass_through.mms
- ub92_v6_0_pass_through.mms

hipaa_x12_4010_pass_through.mms

The **hipaa_x12_4010_pass_through.mms** map validates HIPAA X12 transmission data against the **Partner X12 Inbound Transmission EDI** and **Partner X12 Outbound Transmission EDI** definitions in the **hipaa_x12_4010.mtt** type tree.

This map also includes some pre-defined data audit log settings that can be left as-is or modified and then activated by setting **Data SettingsAudit** in **MapAudit** to **ON**.

cms_276i(_4010a1)_flat_pass_through.mms

The **cms_276i_flat_pass_through.mms** map validates CMS institutional claim status request (276I) flat file data against the **cms_276i_flat file data** definition in the **cms_276i_277i_flat.mtt** type tree.

cms_276p(_4010a1)_flat_pass_through.mms

The **cms_276p_flat_pass_through.mms** map validates CMS professional claim status request (276p) flat file data against the **cms_276p_flat file data** definition in the **cms_276p_277p_flat.mtt** type tree.

cms_277i(_4010a1)_flat_pass_through.mms

The **cms_277i_flat_pass_through.mms** map validates CMS institutional claim status response (277I) flat file data against the **cms_277i_flat file data** definition in the **cms_276i_277i_flat.mtt** type tree.

cms_277p(_4010a1)_flat_pass_through.mms

The **cms_277p_flat_pass_through.mms** map validates CMS professional claim status response (277P) flat file data against the **cms_277p_flat file data** definition in the **cms_276p_277p_flat.mtt** type tree.

cms_835i(_4010a1)_flat_pass_through.mms

The **cms_835i_flat_pass_through.mms** map validates CMS (formerly HCFA) institutional claim payment (835I) flat file data against the **cms_835i_flat file data** definition in the **cms_835i_flat.mtt** type tree.

cms_835p(_4010a1)_flat_pass_through.mms

The **cms_835p_flat_pass_through.mms** map validates CMS (formerly HCFA) professional claim payment (835P) flat file data against the **cms_835p_flat file data** definition in the **cms_835p_flat.mtt** type tree.

cms_837i(_4010a1)_flat_pass_through.mms

The **cms_837i_flat_pass_through.mms** map validates CMS (formerly HCFA) institutional claim and coordination of benefits (837I) flat file data against the **cms_837i_flat file data** definition in the **cms_837i_flat.mtt** type tree.

cms_837p(_4010a1)_flat_pass_through.mms

The **cms_837p_flat_pass_through.mms** map validates CMS (formerly HCFA) professional claim and coordination of benefits (837P) flat file data against the **cms_837p_flat file data** definition in the **cms_837p_flat.mtt** type tree.

nsf_claim_v3_01_pass_through.mms

The **nsf_claim_v3_01_pass_through.mms** map validates NSF Version 3.01 claim and coordination of benefits data against the **FileGrp structure NSF_Claim** definition in the **nsf_claim_v3_01.mtt** type tree.

ub92_v6_0_pass_through.mms

The **ub92_v6_0_pass_through.mms** map validates UB-92 Version 6.0 claim and coordination of benefits data against the **file structure ub92** definition in the **ub92_v6_0.mtt** type tree.

Chapter 5. Sample Data

Sample data is provided for use with the type trees and maps in the Packs for Healthcare.

Overview of sample data

The data files either contain transaction data (in one or more data exchange formats) or application data used by the Compliance Check application for HIPAA X12 data validation.

In this package, there is sample data for HIPAA X12, CMS, and compliance checking. Compliance checking data files are discussed in the Compliance Checking documentation.

Transaction data

The Healthcare Packs include sample transaction data for HIPAA, NCPDP, and CMS.

HIPAA X12

The HIPAA X12 sample data includes transmission data for the finalized HIPAA X12 transaction set definitions. This sample data is taken from the HIPAA Implementation Guides published by Washington Publishing Company. This data has been slightly modified as the original data was found to be non-compliant. There is one file per version or industry code and one file containing all of the sample data. The HIPAA X12 sample data files include the following files:

- 270_and_271_4010a1_ig_examples.dat
- 270_and_271_ig_examples.dat
- 275_4050_ig_examples.dat
- 277_4050_ig_examples.dat
- 276_and_277_4010a1_ig_examples.dat
- 276_and_277_ig_examples.dat
- 278_4010a1_ig_examples.dat
- 278_ig_examples.dat
- 820_4010a1_ig_examples.dat
- 820_ig_examples.dat
- 834_4010a1_ig_examples.dat
- 834_ig_examples.dat
- 835_4010a1_ig_examples.dat
- 835_ig_examples.dat
- 837d_4010a1_ig_examples.dat
- 837d_ig_examples.dat
- 837i_4010a1_ig_examples.dat
- 837i_ig_examples.dat
- 837p_4010a1_ig_examples.dat
- 837p_ig_examples.dat

© Copyright IBM Corp. 2006

- all_hipaa_4010a1_ig_transaction_sets.dat
- all_hipaa_ig_transaction_sets.dat
- nsf_claim_v3_01.dat
- ub92_v6_0.dat

CMS sample data

The CMS sample data includes transmission data for the eight flat file interface data formats associated with the HIPAA 4010 transaction definitions, the finalized addenda definitions, and the most recent versions of the UB-92 and NSF claim formats. This sample data is based on transformations of the HIPAA X12 transaction sets data included with the IBM WebSphere Transformation Extender Pack for HIPAA EDI.

- cms_277i(_4010a1)_flat.dat
- cms_277p(_4010a1)_flat.dat
- cms_276i(_4010a1)_flat.dat
- cms_276p(_4010a1)_flat.dat
- cms_835i(_4010a1)_flat.dat
- cms_835p(_4010a1)_flat.dat
- cms_837i(_4010a1)_flat.dat
- cms_837p(_4010a1)_flat.dat

Chapter 6. Compliance checking

IBM's Compliance Check application for HIPAA X12 data validation and compliance reporting contains a series of executable maps used to validate HIPAA X12 transmission data and report compliance with industry requirements. The **compliance_check.mms** file is the key source file for this application.

The Compliance Check application uses the types of HIPAA data transmission testing documented in the ″Transaction Compliance and Certification″ white paper developed by the Workgroup for Electronic Data Interchange (WEDI) Strategic National Implementation Process (SNIP) Transactions Work Group - Testing Sub Work-Group; dated June 11, 2002.

The Compliance Check application in the Pack for HIPAA EDI supports data validation and compliance reporting for Types 1 through 7.

Types 1 through 7 of HIPAA data transmission testing are defined as follows:

-
- Type 1 - EDI Standard Integrity Testing
- Type 2 - HIPAA Implementation Guide Requirement Testing
- Type 3 - HIPAA Balance Testing
- Type 4 - HIPAA Inter-Segment Situation Testing
- Type 5 - HIPAA External Code Set Testing
- Type 6 - Product Type / Type of Service Testing
- Type 7 - Trading Partner-Specific Testing

WEDI/SNIP HIPAA transaction compliance types

Type 1 - EDI standard integrity testing

To validate basic syntactical integrity of the EDI submission, Type 1 conducts testing of the EDI file for the following:

- valid segments
- segment order
- element attributes
- testing for numeric values in numeric data elements
- validation of X12 or NCPDP syntax
- compliance with X12 rules
- Type 1 exceptions identified by the HIPAA X12 Compliance Check application are reported using the X12 Standard 997 functional acknowledgement and/or 999 implementation transaction set, or TA1 interchange acknowledgement segment.

Type 2 - HIPAA implementation guide requirement testing

Type 2 involves testing for HIPAA implementation guide-specific syntax requirements, such as:

- limits on repeat counts
- used and not used qualifiers

- codes
- elements and segments

There is also testing for:
- HIPAA required or intra-segment situational data elements
- non-medical code sets as laid out in the implementation guide
- values and codes noted in the implementation guide (by means of an X12 code list or table)
- HL, LX, and ENT sequencing
- HL parent-child relationships

Type 2 exceptions identified by the HIPAA X12 Compliance Check application are reported using the X12 Standard 997 functional acknowledgement and/or 999 implementation transaction set, or TA1 interchange acknowledgement segment.

Type 3 - HIPAA balance testing

Type 3 involves testing a transaction for the following:
- balanced field totals
- financial balancing of claims or remittance advice
- balancing of summary fields (if appropriate)

For example, all claim line item amounts equal to the total claim amount.

Type 3 exceptions identified by the HIPAA X12 Compliance Check application are reported using the 824 application advice and/or the X12 standard 999 implementation acknowledgement transaction set.

Type 4 - HIPAA inter-segment situation testing

Type 4 involves testing of specific inter-segment situations described in the HIPAA implementation guides, such that: if A occurs then B must be populated. This is considered to include the validation of situational fields given values or situations present elsewhere in the file. For example, if the claim is for an accident, the accident date must be present.

If the **XML_Validation** parameter is set to **X** or **S**, validation of the XML data in the BIN segment of the 4050 275 (Additional Information to Support a Healthcare Claim) is supported under Type 4 testing.

Type 4 exceptions identified by the HIPAA X12 Compliance Check application are reported using the 824 application advice and/or the X12 standard 999 implementation acknowledgement transaction set.

XML validation is performed for claims attachments in a BIN segment in the 4050 275 transaction (Additional Information to support a healthcare claim or Encounter) if XML _Validation is set to either X or S.

Type 5 - HIPAA external code set testing

Type 5 involves testing for valid implementation guide-specific code set values and other code sets adopted as HIPAA standards such as ICD-9-CM or NDC code sets.

Type 5 exceptions identified by the HIPAA X12 Compliance Check application are reported using the X12 Standard 997 functional acknowledgement and/or 999 implementation transaction set.

Type 6 - product type / type of service testing

Type 6 testing ensures that the segments (records) of data that differ based on certain healthcare services are properly created and processed into claims data formats. Type 6 requirements are described in the Implementation Guides for the different product types, or lines of service. For example, such services as ambulance, chiropractic, podiatry, home health, durable medical equipment, psychiatry, and others have specific requirements in the Implementation Guides. These requirements have to be tested before the transaction is placed in production.

Type 6 exceptions identified by the HIPAA X12 Compliance Check application are reported using the 824 application advice and/or the X12 standard 999 implementation acknowledgement transaction set.

Type 7 - trading partner-specific testing

The Implementation Guides contain some HIPAA requirements that are specific to Medicare, Medicaid, and Indian Health. Compliance or testing with these payer specific requirements is not required from all trading partners. If the trading partner candidate intends to exchange transactions with one of these Implementation Guide special payers, Type 7 testing is required.

Type 7 exceptions identified by the HIPAA X12 Compliance Check application are reported using the 824 application advice and/or the X12 standard 999 implementation acknowledgement transaction set.

Compliance check functionality

The primary objective of the Compliance Check application is to validate HIPAA X12 transmission data against the compliance rules documented in the HIPAA X12 Implementation Guides and to generate one or more acknowledgement responses that explicitly identify the nature and the location of all exceptions within the transmission.

The Compliance Check application can also generate a report to translate the acknowledgement responses generated in order to facilitate error identification.

The Compliance Check application includes the following:

- Support for interchange (ISA/IEA) envelope checking with X12 standard and/or HIPAA-specific TA1 response and specific interchange note codes.
- Support for functional group (GS/GE) and transaction set (ST/SE) envelope checking with subsequent reporting on any or all of the following responses:
  - 4010 X12 standard 997
  - 4010 HIPAA-specific 997
  - 5010 X12 standard 999
- Checks for unrecognized and undefined segment identifiers with subsequent reporting on any or all of the following responses:
  - 4010 X12 standard 997
  - 4010 HIPAA-specific 997
  - 5010 X12 standard 999

- Transaction set structure checking including checks for missing mandatory segments and for segments and loops that exceed maximum occurrence limits with subsequent reporting on any or all of the following responses:
  - 4010 X12 standard 997
  - 4010 HIPAA-specific 997
  - 5010 X12 standard 999
- Transaction set segment and element checking with subsequent reporting on any or all of the following responses:
  - 4010 X12 standard 997
  - 4010 HIPAA-specific 997
  - 5010 X12 standard 999
- Optional transaction set checking for balancing, inter-segment situation, type-of-service and trading partner-specific requirements with subsequent reporting on any or all of the following responses:
  - 4010 Industry specific 824
  - 4050 HIPAA-specific 824
  - 5010 X12 standard 999
- Support for the 999 acknowledgement is available from the Compliance Check application and from IBM Trading Manager for reporting the results of HIPAA X12 Implementation Guide Type 2, 3, 4, 5, 6, and 7 checks.

The type of acknowlegement is selectable. See the table in the ″Compliance Check external parameter file″ for a list of all compliance check parameters.

The HIPAA X12 Implementation Guides and related Table Data available from Washington Publishing Company and the Data Interchange Standards Association (DISA) are the sources of HIPAA X12 compliance testing information. In the event that the information from these two sources is contradictory, IBM considers the information in the published specification and Implementation Guide documents to be the authority.

The Compliance Check application includes support for all finalized HIPAA X12 4010 transaction set standards, HIPAA X12 4010 addenda transaction set definitions, and the 4050 275 and 277 transaction set definitions.

Optimizing performance

The Compliance Check application can be configured to optimize validation efficiency as dictated by the needs and processes of your organization, as well as the structure of your incoming data. It includes an additional component that can be viewed as a preliminary validation component. The preliminary validation component is used to determine which transactions are valid or invalid, but does not pinpoint the specific error or errors. Subsequently, the Compliance Check application determines exactly where and what the errors are, and reports accordingly. The key to optimizing performance is that the application can be configured so that ONLY the invalid transactions are passed to this later processing.

The first component of the Compliance Check application uses the following type trees for pass through:

- **hipaa_x12_4010_type_1.mtt** - type 1 standard X12 validation
- **hipaa_x12_4010_type_2.mtt** - type 2 HIPAA validation

- **hipaa_x12_4010a1_type_2.mtt** - type 2 HIPAA addenda validation
- **hipaa_x12_4010_type_4.mtt** - type 3/4 HIPAA validation
- **hipaa_x12_4010a1_type_4.mtt** - type 3/4 HIPAA addenda validation

The appropriate pass through map is run for each functional group, and a determination is made as to which transactions are valid and which are invalid.

The Compliance Check application is configured using the Validation_Method parameter in the **compliance_check_parameter.dat** file. Values are Map, (the default), Compliance, or Both. This setting determines how input data will be validated. This option set to **M**[ap], is the recommended option for performance. You should, however, refer to the subsequent table on the parameter definition for details.

Also important in performance consideration are the following parameters. In addition to the general information given in the parameter definition table (see the ″Compliance Check external parameter file), the following guidelines should be considered:

- **997_Detail_Max** - Used to determine when to stop producing detailed (AK3/AK4) error information in the standard X12 997 transaction. The default is 9999, if Type 1 Checking is selected (Type 1 checking is not turned on by default), and there is a large number of Type 1 errors in a transaction, this number can be reduced to help increase performance.
- **HIPAA_997_Detail_Max** - Used to determine when to stop producing detailed (AK3/AK4) error information in the HIPAA 997 transaction. The default is 9999, if Type 2 checking is selected (default setting), and there is a large number of Type 2 errors in a transaction, this number can be reduced to help increase performance.
- **HIPAA_824_Detail_Max** - Used to determine when to stop producing detailed 824 error information for a transaction. The default is 9999, if Type 3/4 checking is selected (default setting) and there is a large number of Type 3 or 4 errors in a single transaction, this number can be reduced to help increase performance.
- **HIPAA_999_Detail_Max** - Used to determine when to stop producing detailed 999 error information for a single transaction. If the number of error lines of 999 errors in the received transaction exceeds this limit, then the logic to generate the detailed error information in the IK3/IK4 loop will be bypassed. As the number of errors for a single transaction increases, the processing time required to detect and report these also increases. Therefore, increasing this limit will decrease performance when processing transactions containing numerous errors.

Platform support and configuration

Utility modules

The utility modules are used by the Compliance Check application for HIPAA X12 data structure validation. The following operating system-specific structure validation utility modules are included with the Pack for HIPAA EDI:

- **hcsvu.dll**
- **aix\hcsvu.so**
- **hp\hcsvu.sl**
- **mvs\hcsvu.loadlib**
- **sun\hcsvu.so**
- **linux\hcsvu.so**

The utility modules are organized by operating system in the **platform_support** directory. You must move the applicable utility to the platform to which you are deploying maps. The utility module files are located in the following location: *install_dir* **\packs\healthcare_v***n.n***\hipaa\platform_support**

Versioning has been included to the **hcsvu** exit in the Compliance Check application. In the functional map, **f_CC_Debug_n_Param**, a call to the exit is performed in order to ensure the compatibility with the current version of the Compliance Check application.

If the Compliance Check application fails with the message ″Fail function aborted map″, it may be due to one of the following:

- **hcsvu** exit not found in the **Base_Exit_Directory** (defaults to system library path).
- **hcsvu** exit found but it is an incompatible version.
- actual message found in **compliance_check.log** in the system library path.

Platform specific configuring for z/OS

This section provides sample JCL code to assist with executing the Compliance Check application in the Healthcare Pack for HIPAA EDI.

```
//***********************************************************
//**
//**      RUN COMPLIANCE CHECKER
//** This is some sample JCL to assist with executing the
//** Compliance Check application in the Healthcare Pack
//** for HIPAA EDI.
//** Please refer to Command Server Guide for Z/OS Batch
//** for further guidelines and examples.
//** The program to be executed is DSTX - the command line
//** is passed through the PARM statement which references
//** ddname PARMS - the dataset which contains the commands
//** required to run the compliance check application.
//***********************************************************
//CCDSTX42 EXEC PGM=DSTX,REGION=0M,PARM='-@PARMS'
//***********************************************************
//**
//** STEPLIB identifies the load library where Transformation
//** Extender is installed.
//** If XML Validation is to be done by the compliance check
//** application - an additional steplib may need to be added -
//** the load library where the XML4C is installed
//**
//***********************************************************
//STEPLIB  DD  DSN=MY.LOADLIB,DISP=SHR
//         DD  DSN=XML.LOADLIB,DISP=SHR
//***********************************************************
//** The PARMS ddname references the dataset which contains
//** the commands required to run the compliance check
//** application.
//** Example of the commands is:
//**  DSTXMAP(CCDSTX42) -IF1 CCPARM -IF2 X12IN -IF3 STRUCT
//**  -IF4 QUALIFER -IF5 TYPE6 -IF6 TYPE7 -IF7 TACKXREF
//**  -OF1 CCOUT -OF2 SUMOUT -OF3 OUT997  -OF4 OUT997H
//**  -OF5 TA1OUT -OF6 TA1OUTH -OF7 VALID -OF8 INVALID
//**  -OF9 OUT824H -OF10 OUT999H -OF11 OUTTACK -OF12 OUTHTML
//**  /VX15 CCPARM /VX15 X12IN /VX15 STRUCT /VX15 QUALIFER
//**  /VX15 TYPE6 /VX15 TYPE7 /VX15 TACKXREF
//**  /VX15 CCOUT /VX15 SUMOUT /VX15 OUT997 /VX15 OUT997H
//**  /VX15 TA1OUT /VX15 TA1OUTH /VX15 VALID /VX15 INVALID
//**  /VX15 OUT824H /VX15 OUT999H /VX14 OUTTACK /VX15 OUTHTML
//***********************************************************
```

```
//*
//PARMS    DD  DSN=MY.COMMAND,DISP=SHR
//*
//************************************************************
//** MAIN COMPLIANCE CHECK MAP
//************************************************************
//**
//DSTXMAP  DD  DSN=MY.MAPLIB(MAPTORUN),DISP=SHR
//**
//************************************************************
//**
//** All of the RUN maps executed via the RUN function within
//** the compliance check application need to have a ddname
//** and the name of the map to run.  The ddname needs to be
//** equivalent to the first eight characters in the run map
//** name.
//** The run maps that need ddnames are as follows
//**
//** X12ADDNUMBER RUN MAP
//** X12ALLACKGENERATE RUN MAP
//** X12AUDITREFORMAT RUN AMP
//** X12HIPAA2HTML RUN MAP
//** X12INITIALCONTROLSUMMARY RUN MAP
//** X12SEPARATORCHECK RUN MAP
//** X12SUBELEMENTSEPARATORCHECK RUN MAP
//** X12T1PRECONTENTCHECK RUN MAP
//** X12T2A1PRECONTENTCHECK RUN MAP
//** X12T2PRECONTENTCHECK RUN MAP
//** X12T4A1PRECONTENTCHECK RUN MAP
//** X12T4PRECONTENTCHECK RUN MAP
//** X12T4XMLPRECONTENTCHECK RUN MAP
//** X12TERMINATORCHECK RUN MAP
//** X12TYPE6QUALIFIER RUN MAP
//** X12XMLERRORREFORMAT RUN MAP
//** X4010HIPAASEGMENTDATAAUDIT RUN MAP
//**
//************************************************************
//X12ADDNU DD  DSN=MY.MAPLIB(X12ADDNU),DISP=SHR
//X12ALLAC DD  DSN=MY.MAPLIB(X12ALLAC),DISP=SHR
//X12AUDIT DD  DSN=MY.MAPLIB(X12AUDIT),DISP=SHR
//X12HIPAA DD  DSN=MY.MAPLIB(X12HIPAA),DISP=SHR
//X12INITI DD  DSN=MY.MAPLIB(X12INITI),DISP=SHR
//X12SEPAR DD  DSN=MY.MAPLIB(X12SEPAR),DISP=SHR
//X12SUBEL DD  DSN=MY.MAPLIB(X12SUBEL),DISP=SHR
//X12T1PRE DD  DSN=MY.MAPLIB(X12T1PRE),DISP=SHR
//X12T2A1P DD  DSN=MY.MAPLIB(X12T2A1P),DISP=SHR
//X12T2PRE DD  DSN=MY.MAPLIB(X12T2PRE),DISP=SHR
//X12T4A1P DD  DSN=MY.MAPLIB(X12T4A1P),DISP=SHR
//X12T4PRE DD  DSN=MY.MAPLIB(X12T4PRE),DISP=SHR
//X12T4XML DD  DSN=MY.MAPLIB(X12T4XML),DISP=SHR
//X12TERMI DD  DSN=MY.MAPLIB(X12TERMI),DISP=SHR
//X12TYPE6 DD  DSN=MY.MAPLIB(X12TYPE6),DISP=SHR
//X12XMLER DD  DSN=MY.MAPLIB(X12XMLER),DISP=SHR
//X4010HIP DD  DSN=MY.MAPLIB(X4010HIP),DISP=SHR
//************************************************************
//**
//** Each of the output cards from the run map
//** X12INITALCONTROLSUMMARY which are set to !Create need
//** a DDNAME.  The DDNAME needs to be the first eight
//** characters of the output file name.  The cards
//** requiring a DDNAME are:
//**   Output #1, #2, #3, #4, #6, #8, #10, #13
//**
//************************************************************
//SUMMARY1 DD  DUMMY
//ENVELOPE DD  DUMMY
//CONTENT1 DD  DUMMY
```

```
//CONTENT2 DD  DUMMY
//SUMMARY2 DD  DUMMY
//VALID1   DD  DUMMY
//VALID2   DD  DUMMY
//INVALID  DD  DUMMY
//*
//************************************************************
//**
//** Loadlib where hcsvu.loadlib is located
//**
//************************************************************
//*
//HCSVU    DD  DSN=MY.HCSVU.LOADLIB(HCSVU),DISP=SHR
//************************************************************
//**
//** The input files are listed with the DDNAMES as defined
//** in the command file.
//**
//************************************************************
//CCPARM   DD  DSN=MY.INPUT.PARM,DISP=SHR
//X12IN    DD  DSN=MY.INPUT.X12IN,DISP=SHR
//STRUCT   DD  DSN=MY.INPUT.STRUCT,DISP=SHR
//QUALIFER DD  DSN=MY.INPUT.QUALIFER,DISP=SHR
//TYPE6    DD  DSN=MY.INPUT.TYPE6,DISP=SHR
//TYPE7    DD  DSN=MY.INPUT.TYPE7,DISP=SHR
//TACKXREF DD  DSN=MY.INPUT.TACKXREF,DISP=SHR
//************************************************************
//**
//** If XML validation is being performed on the contents of
//** the BIN segment in a 4050 275 transaction set and the
//** schemas are defined by a ddname in the xml - then the
//** DDNAMES must point to the location of the schemas.
//**
//************************************************************
//LEVELONE DD  DSN=MY.SCHEMA.LEVELONE.XSD,DISP=SHR
//HEADER#1 DD  DSN=MY.SCHEMAL.HEADER#1.XSD,DISP=SHR
//V3DT#1 DD    DSN=MY.SCHEMA.V3DT#1.XSD,DISP=SHR
//*
//************************************************************
//**
//** The output files are listed with the DDNAMES as defined
//** in the command file.
//**
//************************************************************
//*
//CCOUT    DD  DSN=MY.OUTPUT.CCOUT,DISP=OLD
//SUMOUT   DD  DSN=MY.OUTPUT.SUMOUT,DISP=OLD
//OUT997   DD  DSN=MY.OUTPUT.X997,DISP=OLD
//OUT997H  DD  DSN=MY.OUTPUT.X997H,DISP=OLD
//TA1OUT   DD  DSN=MY.OUTPUT.TA1,DISP=OLD
//TA1OUTH  DD  DSN=MY.OUTPUT.TA1H,DISP=OLD
//VALID    DD  DSN=MY.OUTPUT.VALID,DISP=OLD
//INVALID  DD  DSN=MY.OUTPUT.INVALID,DISP=OLD
//OUT824H  DD  DSN=MY.OUTPUT.X824H,DISP=OLD
//OUT999H  DD  DSN=MY.OUTPUT.X999H,DISP=OLD
//OUTTACK  DD  DSN=MY.OUTPUT.TACK,DISP=OLD
//OUTHTML  DD  DUMMY
//*
//************************************************************
//**
//** The log and trace files can be defined for debugging
//** purposes.
//**
//************************************************************
//MERCLOG  DD  SYSOUT=*
//RUNLOG01 DD  SYSOUT=*
//RUNLOG02 DD  SYSOUT=*
```

```
//RUNTRC01 DD  DUMMY
//RUNTRC02 DD  DUMMY
//RUNAUD01 DD  SYSOUT=*
//DSTXDEBG DD  SYSOUT=*
//DSTXTRCE DD  SYSOUT=*
//DSTXAUD  DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//CEEDUMP  DD  DUMMY
//*
//************************************************************
//**
//** Static temporary dataset allocations
//**
//************************************************************
//SYSTMP01 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP02 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP03 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP04 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP05 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP06 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP07 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP08 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP09 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP10 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP11 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP12 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP13 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP14 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP15 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP16 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP17 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP18 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP19 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP20 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP21 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP22 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP23 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//              UNIT=VIO,
//              SPACE=(CYL,(25,25),RLSE)
//SYSTMP24 DD  DSN=&,
//              DISP=(NEW,DELETE,DELETE),
//              DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP25 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP26 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP27 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP28 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
//SYSTMP29 DD  DSN=&,
//             DISP=(NEW,DELETE,DELETE),
//             DCB=(RECFM=FBS,LRECL=80),
//             UNIT=VIO,
//             SPACE=(CYL,(25,25),RLSE)
```

Compliance check objects

The following objects in the Pack for HIPAA EDI are for the Compliance Check application.

Type Trees:

- compliance_check.mtt
- hipaa_x12_4010_segment.mtt
- x12_segment_data_audit.mtt
- x12_skeleton.mtt
- hipaa_x12_4010_type_1.mtt
- hipaa_x12_4010_type_2.mtt
- hipaa_x12_4010a1_type_2.mtt
- hipaa_x12_4010_type_4.mtt
- hipaa_x12_4010a1_type_4.mtt

Data Files

- compliance_check_parameter.dat
- hipaa_x12_4010_structure.dat
- hipaa_x12_4010_qualifier.dat
- hipaa_x12_type_6_value.dat
- hipaa_x12_type_7_value.dat
- tack_codetable.dat

Map Source

- compliance_check.mms

dll

- hcsvu.dll

Note: Because the objects listed above are used in the application, making changes might cause them to work inconsistently or stop working altogether. For best results do not modify the Compliance Check application objects.

Type trees for compliance check maps

Type trees used by the compliance check maps include the following:

compliance_check.mtt

This tree is used by the executable maps in **compliance_check.mms**. The top level groups in the **compliance_check.mtt** include the following:

- **Compliance_Check_Results**
- **Error_Response**
- **Param**
- **Qualifer_Table**
- **Segment_Error**
- **Structure_Detail**
- **TAck_User_Text**
- **Type_6_Xref**
- **Type_7_Xref**
- **X12_Control_Summary**

hipaa_x12_4010_segment.mtt

The **hipaa_x12_4010_segment.mtt** type tree contains definitions of the HIPAA X12 segments that are used by the Compliance Check application for HIPAA X12 data segment validation.

This type tree was developed using the segment definitions from the **hipaa_x12_4010.mtt** type tree. **Input All EDI** is the top-level group.

This type tree is used by the **x4010hipaasegmentdataaudit.mmc** executable map to generate a data audit log containing details about errors at the element and sub-element levels as well as violations of intra-segment, situational requirements. (The source for the executable map is located in **compliance_check.mms.**)

x12_segment_data_audit.mtt

The **x12_segment_data_audit.mtt** type tree contains a definition of the IBM audit log that is used in the Compliance Check application for HIPAA X12 data validation.

This type tree is a customized version of IBM's standard audit log type tree. It has been customized to focus on the X12 Segment and Element level details in the data audit log results from running the **x4010hipaasegmentdataaudit**.

X12_Segment_Detail Specific Audit_Log Data is the top-level group in this type tree.

This type tree is used by the **x12auditreformat.mmc** executable map to read in the contents of the data audit log generated by the **x4010hipaasegmentdataaudit.mmc** executable map. (The source for both executable maps is located in **compliance_check.mms**.)

x12_skeleton.mtt

The **x12_skeleton.mtt** type tree contains generic definitions of X12 data that are used in the Compliance Check application for HIPAA X12 data validation.

Top-level groups:

- **X12_skeleton File EDI**
- **Partner X12 Outbound Transmission EDI**

X12_skeleton File EDI defines an X12 transmission in such a way that detailed ISA and IEA; GS and GE; and ST and SE checking can be performed. This type tree is used to identify the segments within a transaction set without actually defining the segment details. This capability is used in the Compliance Check application to pass transaction set data to maps that perform transaction set level compliance checking.

Partner X12 Outbound Transmission EDI is used to construct outbound 997; 997H; 824H; 999H and TA1 transmissions.

The executable maps in **compliance_check.mms** use this type tree.

hipaa_x12_4010_type_1.mtt

The **hipaa_x12_4010_type_1.mtt** type tree is used for X12, Type 1 checking only. The top level group is **HeaderInfo**. This type tree is used by **x12t1precontentcheck.mmc** pass-through validation in the Compliance Check application.

hipaa_x12_4010_type_2.mtt

The **hipaa_x12_4010_type_2.mtt** type tree performs two type 2 checks, plus a type 1, X12 check. The top level group is **HeaderInfo**. The type tree is used in **x12t2precontentcheck.mmc** for type 2 pass-through validation.

hipaa_x12_4010a1_type_2.mtt

The **hipaa_x12_4010a1_type 2.mtt** type tree performs type 2 addenda checks plus the type 1, X12 check. The top level group is **HeaderInfo**. This type tree is used by **x12t2a1precontentcheck.mmc** for type 2 addenda pass-through validation.

hipaa_x12_4010_type_4.mtt

The **hipaa_x12_4010_type_4.mtt** type tree provides type 3 and 4 checking plus type 2 and type 1, x12 checking. The top level group is **HeaderInfo**. This type tree is used by **x12t4precontenecheck.mmc** for type 4 pass-through validation.

hipaa_x12_4010a1_type_4.mtt

The **hipaa_x12_4010a1_type_4.mtt** type tree provides type 3 and 4 addenda checking plus type 2 addenda checking and type 1, X12 checking. The top level group is **HeaderInfo**. This type tree is used in **x12t4a1precontentcheck.mmc** for type 4 addenda pass-through validation.

Data files

This section discusses the application data for healthcare application maps for HIPAA compliance checking.

The HIPAA X12 Compliance Check application data includes five files required as input for the Compliance Check application.

The following HIPAA X12 Compliance Check application data files are included with the Pack for HIPAA EDI:

- compliance_check_parameter.dat
- hipaa_x12_4010_structure.dat
- hipaa_x12_4010_qualifier.dat
- hipaa_x12_4010_type_6_value.dat
- hipaa_x12_4010_type_7_value.dat
- tack_codetable.dat

compliance_check_parameter.dat

The **compliance_check_parameter.dat** data file contains parameter settings that are used by the Compliance Check application for HIPAA X12 data validation. This data file is used as the default input for card 1 in the **compliance_check.mms** map. This data file, or a modified copy, must be ported to the target platform along with the compiled maps for the Compliance Check application.

The settings in this file can be edited and will take effect the next time that the Compliance Check application is executed. There is no need to rebuild or re-deploy maps. This file contains data records that adhere to the convention of Tag_Name=″Value″.

The following table describes the parameter tag names and values used by the Compliance Check application in the Pack for HIPAA EDI.

Parameters are defined as tag=″setting″ pairs. Each parameter is preceded by a narrative comment block. The slash-star characters indicate the start and the star-slash characters indicate the end of the comment block. If the parameter is to be set to the default value there is no need to explicitly define the setting and, for performance reasons, the parameter should be commented out using a semi-colon (;) as the first character in the line.

Compliance check external parameter file

In the following table, the * (asterisk) indicates the default value

Parameter Tag Name	Values	Description
Product_Name	**H**[CIP]* CM	The **Product_Name** parameter is used to identify the IBM product that has invoked the Compliance Check application.

Parameter Tag Name	Values	Description
Validation_Method	**M**[AP]* **C**[OMPLIANCE] **B**[OTH]	**M** - This default setting sends all data through the pass through validation maps and only the invalid data goes through the subsequent processing. This is the recommended setting. **C** - This setting causes all data to be passed to the detailed processing done by the Compliance Check application. Pass through validation is not executed. **B** - This setting causes all data to be processed both by the pass through validation maps and the detailed compliance check validation. If there is a large amount of data, this option can slow the process. This option should be used for troubleshooting purposes, logic module, or both.
Operating_System	**H**[PUX] **S**[UNOS] **A**[IX] No default setting.	Overrides the dynamic determination of the UNIX operating system from the format of the input data file reference, or from the execution of a shell command, GET("SHL","-CMD uname"). There is NO default setting for this parameter. On UNIX, if the parameter is not specified, the operating system will be determined dynamically at map execution time. On Windows or z/OS, the operating system is always determined dynamically.
Ignore_Whitespace	**Y**[ES] - Ignore whitespace characters **N**[O] - Do not ignore whitespace characters*	Specifies whether whitespace characters (i.e. <SP>, <NL>, <CR>, <LF>) that occur either at the end of the data input file or between interchanges should be ignored.
Base_Map_Directory	No default setting.	Specifies the directory for compiled maps that will be executed by **compliance_check.mmc** via the RUN function. There is no default setting for this parameter. If the parameter is not specified, then the system library path will be used.

Parameter Tag Name	Values	Description
Base_Exit_Directory	No default setting.	Specifies the directory for executables invoked via the EXIT function (i.e. **hcsvu**). There is no default setting for this parameter. If the parameter is not specified then the system library path will be used. **Note** The **Base_Exit_Directory** parameter cannot be used on z/OS
Paging_Parameter	64:8	Specifies page size and page count to be used for execution of compiled maps that will be executed by **compliance_check.mmc** via the RUN function. Typical settings include: **64:8** - Page size of 64K and page count of 8 **256:8** - Page size of 256K and page count of 8 Page size value may be an even integer between 2 and 1024. Page count may be an integer between 1 and 9999. In general, larger page size and count settings will improve overall map performance when the input data transmission is large. Determining the optimal page settings may require experimentation. For performance reasons, this parameter setting is not used for compiled maps that operate exclusively on small quantities of data (i.e. **x12terminatorcheck.mmc**). When modifying this parameter, ensure that a space is entered prior to the command.

Parameter Tag Name	Values	Description
Audit_Command	**-A** - Do not generate an audit log **-AE=C:** Execution summary audit created in the specified directory and overwritten each time the map is executed **-AEWU*** - Execution summary audit created on warning or error and with a unique file name.	Specifies the audit log setting to be used for the execution of compiled maps that will be executed by **compliance_check.mmc** via the RUN function.
Work_File_Command	**WDU***	Specifies the audit log setting to be used for the execution of compiled maps that will be executed by **compliance_check.mmc** via the RUN function. When modifying this parameter, ensure that a space is entered prior to the command.
HIPAA_Response_Ack	**P**[RE 999]* - Only 997/824s will be produced for all levels of validation chosen. **A**[FTER 999] - 999 is the only response produced for all levels of validation. **B**[OTH] - 997/824s and 999 produced, both report all levels of validation.	Determines which acknowledgements will be produced by compliance check.
HIPAA_Type_1_Checks_Enabled	**Y**[ES] **-** Perform Type 1 checks on the data. **N**[O]* - Do not perform Type 1 checks on the data.	Used to indicate whether WEDI/SNIP Type 1 checks should be performed on the input data. If you change the default NO setting to YES, you must also change the **997_Enabled** parameter in order to create a 997.
HIPAA_Type_2_Checks_Enabled	**Y**[ES]* **N**[O]	Used to indicate whether WEDI/SNIP Type 2 checks should be performed on the input data.

Parameter Tag Name	Values	Description
HIPAA_Type_3_Checks_Enabled	**Y**[ES]* **N**[O]	Used to indicate whether WEDI/SNIP Type 3 checks should be performed on the input data.
HIPAA_Type_4_Checks_Enabled	**Y**[ES]* **N**[O]	Used to indicate whether WEDI/SNIP Type 4 checks should be performed on the input data.
HIPAA_Type_5_Checks_Enabled	**Y**[ES] **N**[O] *	Used to indicate whether WEDI/SNIP Type 5 checks should be performed on the input data.
HIPAA_Type_6_Checks_Enabled	**Y**[ES] **N**[O] *	Used to indicate whether WEDI/SNIP Type 6 checks should be performed on the input data.
HIPAA_Type_7_Checks_Enabled	**Y**[ES] **N**[O] *	Used to indicate whether WEDI/SNIP Type 7 checks should be performed on the input data.
XML_Validation	**N**[EVER]* **X**[ML] **S** [CHEMA]	Indicates whether the data within the BIN segment of the 4050 275 transaction set is to be validated - and to which level. For either setting X or S, all applicable schemas need to be in the map execution directory.
997_Enabled	**N**[EVER]* **E**[RRORS] **A**[LWAYS]	Used to indicate whether the results from the WEDI/SNIP Type 1 checks should be reported using the X12 standard 997 transaction set.
997_GS08_Setting	**S**[TANDARD]* **I**[NDUSTRY]	Used to indicate whether the GS08 value in the X12 standard 997 should be set to the X12 standard value or to the HIPAA-specific value.
HIPAA_997_ISA_Version	**4010*** - Create the 997 ISA envelope in X12 4010 standards **4050** - Create the 997 ISA envelope in X12 4050 standards	Used to indicate whether the interchange (ISA) envelopes for the 997 acknowledgement should be created in the X12 4010 or 4050 version.
TA1_Enabled	**N**[EVER] **E**[RRORS] **A**[LWAYS]*	Used to indicate whether the results from the WEDI/SNIP Type 1 interchange envelope checks should be reported using the X12 standard TA1 segment.
HIPAA_997_Enabled	**N**[EVER] **E**[RRORS] **A**[LWAYS]*	Used to indicate whether the results from the WEDI/SNIP Type 2 and 5 checks should be reported using the HIPAA specific 997 transaction set.

Parameter Tag Name	Values	Description
HIPAA_997_Reject_Level	**A**[CCEPT]* **R**[EJECT]	Used to indicate whether the HIPAA 997 should be rejected or accepted based on the results of higher levels of validation (T3/T4/T6/T7). This paramater will only have an impact on the AK501 if Type 2 WEDI/SNIP validation passes compliance check, but higher levels of validation fail (Type 3, 4, 6 or 7 WEDI/SNIP validations). It should be noted that if REJECT is selected and there aren't any Type 2 errors, but a higher level of validation contains errors, then there won't be any AK3/AK4 information in the 997 to indicate the cause of failure. If the Reject Level is Type 2, the AK501 will show a status of Accept if Type 2 validation passes compliance check but higher levels fail. If the Reject Level is higher than Type 2 the AK501 will show a status of Reject if Type 2 validation passes compliance check but a higher level of validation fails.
HIPAA_TA1_Enabled	**N**[EVER] **E**[RRORS]* **A**[LWAYS]	Used to indicate whether the results from the WEDI/SNIP Type 3, 4, 6 or 7 checks should be reported using the 824 transaction set.
HIPAA_824_Enabled	**N**[EVER] **E**[RRORS]* **A**[LWAYS]	Used to indicate whether the results from the WEDI/SNIP Type 3, 4, 6 or 7 checks should be reported using the 824 transaction set.
Loop_Id_In_Acknowledgement	**N**[O]* - No value reported. **Y**[ES] - Use the industry-specific value (i.e. 2310C).	Used to indicate whether the loop identifier is required on the acknowledgement. If a 997 is requested and this parameter is set to Yes, then the AK303 will contain the loop identifier of the segment in error. If a 999 is requested and this parameter is set to Yes, then the IK303 will contain the loop identifier of the segtment in error.

Parameter Tag Name	Values	Description
HIPAA_824_GS08_Setting	**S**[TANDARD]* **I**[NDUSTRY]	Used to indicate whether the HIPAA 824 produced will be an industry version (4010 824 Acknowledgement) or the standard version (4050 824 Acknowledgement). When the setting is **S**[TANDARD], the structure and contents of the 824 acknowledgement transmission produced will follow the rules in the 004050 Implementation Guide. When the setting is **I**[NDUSTRY], the structure and contents of the 824 acknowledgement transmission produced will follow the rules in the 004010X161 Implementation Guide.
HIPAA_824_ISA_Version	**4010*** - Create the 824 ISA envelope in X12 4010 standards. **4050** - Create the 824 ISA envelope in X12 4050 standards	Used to indicate whether the interchange (ISA) envelopes for the 824 acknowledgement should be created in the X12 4010 or 4050 version.
HIPAA_999_Enabled	**N**[EVER]* **E**[RRORS] **A**[LWAYS]	Used to indicate whether the results from the WEDI/SNIP checks should be reported using the X12 5010 999 transaction set.
HIPAA_999_ISA_Version	**5010*** - Create the 999 ISA envelope in X12 5010 standards. **4010**- Create the 999 ISA envelope in X12 4010 standards. **4050** - Create the 999 ISA envelope in X12 4050 standards.	Used to indicate whether the interchange (ISA) envelopes for the 999 acknowledgement should be created in the X12 5010, 4010, or 4050 version.
HIPAA_Tack_Enabled	**Y**[ES] - Generate. **N**[O]* - Do not generate.	Determines if the Translated Acknowledgement Report will be produced. This report transforms the 997, 997H, and 824 acknowledgements into a more readable report
HIPAA_Tack_Accepted	**A**[LL]- Generated accepted error report. **E**[RROR]* - Generate error report only.	Determines what level of reporting will appear on the Translated Acknowledgement Report. The options are to produce the report for all data, (accepted and rejected) or to only generate the report based on the data in error.

Parameter Tag Name	Values	Description
Acknowledgment_Type	**S**[EPARATE]* **C**[OMBINED]	Used to specify whether TA1 acknowledgments should be combined with the response transmission that contains the 997 acknowledgment transaction sets or placed in a separate response transmission.
Include_GS06_In_Grouping	**Y**[ES] - Include GS06 in grouping for acknowledgement. **N**[O]* - Do not include GS06 in grouping for acknowledgement. **A**[LWAYS] - Create a func group in acknowledgement based on func groups in input.	Used when it is required to include the functional group control number in the grouping of functional groups within an acknowledgement. If the setting is No, only the GS02, GS03, GS07 and GS08 are used to determine a unique functional group for reporting in the acknowledgement.
997_Detail_Max	9999*	Used to determine when to stop producing detailed 997 error information for a single transaction. If the number of 997 errors in the received transaction exceeds this limit then the logic to generate the detailed error information in the AK3/AK4 loop will be bypassed. As the number of errors for a single transaction increases the processing time required to detect and report these also increases. Therefore, decreasing this limit will improve performance when processing transactions containing numerous errors.
HIPAA_997_Detail_Max	9999*	Used to determine when to should stop producing detailed HIPAA 997 error information for a single transaction. If the number of HIPAA 997 errors in the received transaction exceeds this limit then the logic to generate the detailed error information in the AK3/AK4 loop will be bypassed. As the number of errors for a single transaction increases the processing time required to detect and report these also increases, so increasing this limit will decrease performance when processing transactions containing numerous errors.

Parameter Tag Name	Values	Description
HIPAA_824_Detail_Max	9999*	Used to determine when we should stop producing detailed 824 error information for a single transaction. If the number of 824 errors in the received transaction exceeds this limit then the logic to generate the detailed error information in the TED loop will be bypassed. As the number of errors for a single transaction increases the processing time required to detect and report these also increases, so increasing this limit will decrease performance when processing transactions containing numerous errors.
HIPAA_999_Detail_Max	9999*	Used to determine when to stop producing detailed 999 error information for a single transaction. If the number of error lines of 999 errors in the received transaction exceeds this limit, then the logic to generate the detailed error information in the IK3/IK4 loop will be bypassed. As the number of errors for a single transaction increases, the processing time required to detect and report these also increases, so increasing this limit will decrease performance when processing transactions containing numerous errors.
Duplicate_ISA_Check_Enabled	**Y**[ES]* **N**[O]	Used to indicate whether interchange (ISA) envelopes should be checked for duplicates.
Duplicate_ST_Check_Enabled	**Y**[ES]* - Perform duplicate checking of transaction set envelopes. **N**[O] - Do not perform duplicate checking of transaction set envelopes.	Used to indicate whether transaction set (ST) envelopes should be checked for duplicates.
Duplicate_ST_Reject_Level	**F**[UNCTIONAL GROUP] **T**[RANSACTIO**N** SET]*	Indicates whether or not an entire functional group should be rejected when duplicate ST control numbers exist within the functional group.

Additional notes on parameter file changes

When changing the setting, ensure that the ″**;**″ (semicolon) prior to the tag=″setting″ is removed or the application will not recognize the change.

When changing any of the following parameters, ensure that a space is entered prior to the setting variable. For example: Paging_Parameter=″ -256:8″.

- Paging_Parameter
- Audit_Command
- Work_File_Command

By default, Type 2, 3 and 4 checking are turned on. In order to turn on any additional checks - ensure that it is changed in two places.

For example, to turn on Type 1 checking, set the **HIPAA_Type_1_Checks_Enabled** parameter setting to **Y** then set the **997_Enabled** response parameter accordingly.

When changing any of the directory path settings (Base_Map_Directory, Base_Exit_Directory), Audit directory, Work files directory) and there i a space in the directory name, then the entire directory needs to be contained within double quotes.

hipaa_x12_4010_structure.dat

The **hipaa_x12_4010_structure.dat** data file contains predefined structure details definitions that are used by the Compliance Check application for HIPAA X12 data validation. This data file is used as the default input for card 3 in the **compliance_check.mms** map. This data file must be ported to the target platform along with the compiled maps for the Compliance Check application. Note that when transferring the structure file to z/OS, a data set must be allocated with a record size that is large enough to hold the longest record in the file without wrapping.

hipaa_x12_4010_qualifier.dat

The **hipaa_x12_4010_qualifier.dat** data file contains a list of rules used for performing Type 2, Type 3, and Type 4 validations.

This data file must be ported to the target platform along with the compiled maps when using the Compliance Check application. It should be noted that when transferring to z/OS, a data set must be allocated with a record size that is large enough to hold the longest record in the file without wrapping.

hipaa_x12_type_6_value.dat

The **hipaa_x12_type_6_value.dat** file is a pipe delimited text file that is used in performing Type 6 validation. See ″Compliance Checking.″

hipaa_x12_type_7_value.dat

The **hipaa_x12_type_7_value.dat** file is a pipe delimited text file that is used in performing Type 7 validation. See ″Type 7 Validation.″

tack_codetable.dat

The **tack_codetable.dat** file contains descriptions of errors and their associated codes, element names, sub element names, segment names and composite names which were extracted from the Washington Publishing Company (WPC) tables. The file is used in the Compliance Check application for translating the acknowledgement error codes into a more readable format.

Compliance check maps

The map source file (**compliance_check.mms**) included with the Pack for HIPAA EDI contains several executable maps for compliance checking. All of the maps are described in this section.

Note: The compliance checking maps are used by applications. If you modify them, applications might work inconsistently or stop working altogether.

compliance_check.mms

compliance_check.mms is the map source file that contains all of the executable maps associated with the Compliance Check application, including the top-level map, **compliance_check**.

Executable map descriptions

The following table lists names and descriptions of the executable maps and their associated output cards, contained in **compliance_check.mms**.

You must build all of the executable maps to create the compiled map files (***.mmc**) used by the RUN function calls before you run the **compliance_check.mmc** executable map.

Executable Map	Output Card	Description
compliance_check	1	Top-level, or ″backbone″ map for the HIPAA X12 Data Validation and Compliance Reporting application. This map is responsible for keeping track of all **compliance check** processing steps and generating all application output. Output defaults to the sink.
	2	**compliance_check_summary.out** - contains the compliance summary portion of the results from output card 1
	3	c**ompliance_check_997.out** - contains the X12 997 if type 1 checking is enabled.
	4	**compliance_check_997h.out** - contains the HIPAA 997 if type 2 checking is enabled
	5	**compliance_check_ta1.out** - contains the X12 TA1 results if type 1 checking is enabled
	6	**compliance_check_ta1h.out** - contains the HIPAA TA1 results and type 2 checking is enabled
	7	**compliance_check_valid.out** - contains the valid data as determined by the Compliance Check application
	8	**compliance_check_invalid.out** - contains the invalid data as determined by the Compliance Check application
	9	**compliance_check_824h.out** - contains the type 3 and 4 errors if type 3 and/or 4 checking are enabled.

Executable Map	Output Card	Description
	10	**compliance_check_999h.out** - contains the X12 999 results of all the checks enabled.
	11	**compliance_check_tack.html** - a heirarchial report containing an "English" translation of the responses generated.
	12	**TAck_EDI.html** - reformats the input EDI data to html format and adds segment numbers to each segment.
x12initialcontrolsummary	1	Reads in X12 transmission data; identifies the key interchange, functional group, and transaction set information and uses it to populate an X12 control summary file. Invoked from the **compliance_check** map, the **x12initialcontrolsummary** map is responsible for all compliance check processing except generation of acknowledgements. The 1st output card is responsible for creating X12 control summary information and populating pointers and data fields used throughout the remainder of compliance check processing.
	2	Envelope checking - Allows checking and reporting of interchange, functional group, and transaction set envelope errors as defined by the X12 4010 standard. Envelope checking also updates the control summary file with the results of checks performed on the ISA/IEA, GS/GE, and ST/SE envelope structures and error code details for reporting compliance exceptions.
x12separatorcheck	N/A	Invoked from envelope checking, the **x12separatorcheck** map validates the input data string against the definition of the X12 data element separator and returns a value of `A` if the input is considered a valid X12 data element separator and `R` if it is not.
x12terminatorcheck	N/A	Invoked from envelope checking, the **x12terminatorcheck** map validates the input data string against the definition of the X12 segment terminator and returns a value of **A** if the input is considered a valid X12 segment terminator and **R** if it is not.

Executable Map	Output Card	Description
	3	Preliminary validation checking - Allows preprocessing of the EDI HIPAA X12 input data at the functional group level. Each transaction set in the control summary file is updated with a status of ACCEPT if a transaction set is valid and ERRORS NOTED if it is invalid. This allows bypassing of content check if the data is valid which results in a substantial performance increase when the data is valid.
X12t2precontentcheck	N/A	Invoked from preliminary validation checking, the **x12t2precontentcheck** map is a pass through validation map that validates a functional group at the type 2 level. This map will only be invoked if the highest level of validation chosen is type 2. It returns the transaction set control numbers of any transaction sets in error back to pre content checking.
X12t1precontentcheck	N/A	Invoked from preliminary validation checking, the **x12t1precontentcheck** map is a pass through validation map that validates a functional group at the type 1 level. This map will only be invoked if the highest level of validation chosen is type 1. It returns the transaction set control numbers of any transaction sets in error back to pre content checking.
X12t2a1precontentcheck	N/A	Invoked from preliminary validation checking, the **x12t2a1precontentcheck** map is a pass through validation map that validates an addenda functional group at the type 2 level. This map will only be invoked if the highest level of validation chosen is type 2. It returns the transaction set control numbers of any transaction sets in error back to pre content checking.
X12t4precontentcheck	N/A	Invoked from preliminary validation checking, the **x12t4precontentcheck** map is a pass through validation map that validates a functional group at the type 4 level. This map will be invoked if the highest level of validation chosen is type 3 or 4. It returns the transaction set control numbers of any transaction sets in error back to pre content checking.

Executable Map	Output Card	Description
X12t4a1precontentcheck	N/A	Invoked from preliminary validation checking, the **x12t4a1precontentcheck** map is a pass through validation map that validates an addenda functional group at the type 4 level. This map will only be invoked if the highest level of validation chosen is type 3 or 4. It returns the transaction set control numbers of any transaction sets in error back to pre content checking.
	4	Content checking validates the contents of the transaction set for any transaction the failed pre content checking and updates the control summary file with the results.
xmlt4a1precontentcheck	N/A	Invoked from preliminary validation checking, **x12t4xmlprecontentcheck** is an XML validation map that validates the XML data in the BIN segment of the 4050 275 transaction. It is only invoked if the Validation_Method is M or B and the **XML_Validation** flag is set to **X** or **S**.
x4010hipaasegmentdataaudit	N/A	Invoked from content checking, the **x4010hipaasegmentdataaudit** map checks the transaction set segments for data attributes and element relationships one segment at a time and reports any exceptions.
x12auditreformat	N/A	Invoked from content checking, the **x12auditreformat** map is responsible for reformatting error data from the data audit log produced by **x4010hipaasegmentdataaudit**.
	5	Compliance Summary Start contains the value ″<COMPLIANCE SUMMARY>″ which is used to help parse the data.
	6	Compliance Summary consists of merging the results from all of the validation steps and updates the status indicators in the control summary file.
	7	Compliance Summary End contains the value ″<COMPLIANCE SUMMARY>″ which is used to help parse the data.
	8	Build Valid Transactions extracts the valid transactions as reported in the control summary file.
	9	Valid Transactions Start contains the value ″<VALID TRANSACTIONS>″ which is used to help parse the data.
	10	Valid Transactions extracts all the non empty transactions from the data created in output card 6.
	11	Valid Transactions End contains the value ″<VALID TRANSACTIONS>″ which is used to help parse the data.

Executable Map	Output Card	Description
	12	Invalid Transactions Start contains the value ″<INVALID TRANSACTIONS>″ which is used to help parse the data.
	13	Invalid Transactions extracts the invalid transactions as reported in the control summary file.
	14	Invalid Transactions End contains the value ″<INVALID TRANSACTIONS>″ which is used to help parse the data.
x12xmlerrorreformat.mmc	N/A	Invoked from Type 4 content checking, the **xmlerrorreformat** map is responsible for reformatting any XML error data.
x12allackgenerate	N/A	Invoked from the **compliance_check** map, the **x12allackgenerate** map extracts the acknowledgement details from the X12 control summary file and generates the outbound transmissions containing the transaction sets of the acknowledgements requested.
	1	Contains the outbound transmission of the X12 997.
	2	Contains the outbound transmission of the TA1.
	3	Contains the outbound transmission of the HIPAA specific 997.
	4	Contains the outbound transmission of the 824.
	5	Contains the outbound transmission of the HIPAA specific TA1.
	6	Contains the outbound transmission of the 999.
	7	Contains the sorted outbound transmssion of the 999.
	8	Contains the translated acknowledgement report.
	9	A concatenation of output cards 1-8 and is passed back to the main compliance check map for reporting.
x12hipaa2html	N/A	Is executed from output card number 12 of the main compliance check map. It wraps the input data in HTML format. It has two output cards - one calls the X12addnumber to add segment numbers. The second one wraps and writes the input data as HTML.
x12addnumber	N/A	Is executed from x12hipaa2html and adds segment numbers to the HTML wrapped edi data. The resulting output is passed back to the compliance check map (output card #12) and the file created is Tack_EDI.html. It is created on content.

Executable Map	Output Card	Description
x12t4xmlprecontentcheck.mmc	N/A	Preliminary validation map to validate the xml against the level requested. It will execute if transaction is a 4050 275, XML_Validation is set to ″X″ or ″S″ and Validation_Method is set to ″M″ or ″B″.

Compliance reporting

One or more acknowledgement responses are used for reporting HIPAA X12 data validation results. TA1, 997, 999 and 824 responses are available from the Compliance Check application. The type of acknowledgement responses that are generated for a given set of HIPAA X12 transmission data depends on the following:

- the type of testing being performed
- the existence and location of any data validation exceptions
- the settings of various reporting parameters

The X12 TA1 message is used to report interchange level (ISA/IEA) enveloping errors. All other error conditions are reported using the X12 997, 824 and/or 999 transaction sets.

The Compliance Check application also has the ability to transform the TA1, 997 and 824 responses into a more readable format in order to easily interpret the responses and quickly identify the location of reported problems.

Separate outputs are used for reporting the results from X12 standard data validation and those from HIPAA-specific checks. In the case of the X12 transmission containing 997 transaction sets, the Version / Release / Industry ID Code (GS08) will be used to differentiate between the two. When the 997 acknowledgment response contains the results of the X12 standard checks, the GS08 will be set to `004010`. When the response contains the results of HIPAA-specific checks, the GS08 will be set to the industry-specific code of the form `004010`<*xnnn*>.

Translated acknowledgements

The Compliance Check application has the ability to transform the 997 and the 824 acknowledgments into a more readable report format. The acknowledgement report will display the entire segment in error – not just the data, by linking to an additional output from the Compliance Check application.

The report is only available at the time the compliance check application is executed and is enabled by changing the parameter values described below:

- **HIPAA_Tack_Enabled** - determines if the report is to be produced.
- **HIPAA_Tack_Accepted** - determines the level of reporting.

Color coding is included in the report as follows:

- **Green** - indicates that everything below that level is accepted.
- **Amber** - indicates that something below that level has an error.
- **Red** - indicates that everything below that level has been rejected.

- If the functional group is amber, then at least one transaction set within the functional group was rejected.
- If the Interchange is red, then all the functional groups and all the transaction sets within the interchange have been rejected.
- If the transmission is green, then the entire transmission has been accepted.

The report merges all of the information from the various acknowledgements requested, into one hierarchical report.

Data files

The associated data files are described as follows:

- **tack_codetable.dat** - contains descriptions of errors and their associated codes, element names, sub-element names, segment names and composite names which were extracted from the Washington Publishing Company (WPC) tables.
- **compliance_check_tack.css** - cascading style sheet which is provided to allow modifications to colors, fonts, margins, scroll regions etc.
- **plus.gif** - Used to visually demonstrate a **+** (plus) on the report which indicates the data can be opened.
- **minus.gif** - Used to visually demonstrate a **-** (minus) on the report which indicates the data can be closed.

Note: In order to properly view the report generated, you must have the last 3 files in the same directory as the report.

Chapter 7. Type 5 validation

This documentation describes an example that includes step-by-step instructions, maps, trees, and test cases needed to add the ″X12 Code Source 22: US States and Outlying Areas Code Validation″ to the Packs for Healthcare.

Type 5 validation example

Type 5 validation includes an example of how to perform the validation through a restriction list and a component rule. The restriction list and component rule are populated from a text file.

There is also an example of how to perform the validation using the DBLOOKUP function in the component rule using database drivers. The databases drivers used are:

- MS Access
- MS Excel
- MS Text

The process behind the example map is to export the HIPAA tree(s) being used, modify them with a map to add the validation, import the tree(s), and run the application.

Type 5 validation should only be added to **hipaa_x12_4010_segment.mtt** and **hipaa_x12_4010.mtt** type trees. If Type 5 validation is added to **hipaa_x12_4010_type_1.mtt**, or **hipaa_x12_4010a1_type_1.mtt** then Type 5 will be activated during Type 1 processing which is not consistent with X12 and HIPAA processing guidelines. Before performing the following steps, you should make a copy of the type tree that you are exporting. Copy **hipaa_x12_4010_segment.mtt** into the **type_5_check** example folder and perform the following procedures from that example folder.

The following example uses a single external code data source for ″X12 Code Source 22: US States and Outlying Areas Code Validation″, with added codes for the Canadian provinces, as specified in the HIPAA implementation guides. The following procedures describe how to perform the implementation to insert the validation as a restriction list.

To perform the implementation perform the following steps:

1. Copy the **hipaa_x12_4010_segment.mtt** to the installation directory. By default this directory is:

   install_dir\**packs\healthcare_v***n.n*\**hipaa\examples\type_5_check**
2. Open the type tree in Type Designer.
3. Search for **Unknown_N4** then open the **StateProvinceCd** item to view the restriction list or component rule.

   Note: There are no values in the restriction list. The example map will import values from the **x12_external_code_source_22.txt** file (included in the Pack for HIPAA EDI) into the restriction list or the component rule.
4. Click on the root of the tree and then export the tree from the **Tree** menu. The exported tree will now have an extension of **.mts**.

5. In Map Designer, open the **add_type_5_checks.mms** map included in this package.

   You may need to adjust the **ttmaker** paths in the cards to your install location. **ttmaker** is shipped with the IBM Transformation Extender and can be found by default in:
   *install_dir***\examples\TTMAKER\Tree**
   Or
   *install_dir***\examples\dsgnstud\ttmaker\export**
   Refer to the Type Tree Maker documentation for more information about **ttmaker**.
6. In the **add_type_5_checks** map source file, go to the following executable map: **add_type_5_restriction_list**. There are two input cards and one output card. The first input is the type tree export file that was created in step 4. You must change this card to use that exported **.mts** file. The second input is the **x12_external_code_source_22.txt** data file (included in the Pack for HIPAA EDI). The output will be the updated **.mts** file with the new restriction list added. Adjust the output card to a specific file name for your environment.
7. Build and run the map to create the updated **.mts** file. You may open the input **.mts** and output **.mts** file in any text editor to compare the differences. The updated **.mts** file includes a restriction list of States and outlying areas as follows:

   <ValueRestrictions IgnoreCase=″NO″ Rule=″INCLUDE″><Value description=

   ″ALBERTA″>AB</Value>

   <Value Description=″ALABAMA″>AL</Value>

   <Value Description=″ALASKA″>AK</Value>

   <Value Description=″ARKANSAS″>AR</Value>...
8. The updated **.mts** files must be imported back into a tree. From the Type Designer **Tree** menu, select **Import**.
9. Select Type Tree Maker from Importer Wizard and select the updated **.mts** file created in step 7 to import. This tree, created as a result of the import, is: <*input-tree-name*>**_updated.mtt**
10. Open the newly created type tree and verify that the state restriction lists have been added.
11. Backup the original **hipaa_x12_4010_segment.mtt** file, then copy the updated tree **hipaa_x12_4010_segment_updated.mtt** to:*install_dir***\packs\ healthcare_v***n.n***\hipaa\ trees\hipaa_x12_4010_segment.mtt**
12. Open the **hipaa_x12_4010_segment** tree in the Type Designer and analyze the new segment tree. It should analyze with 0 warnings and 0 errors.
13. Repeat steps 1 through 12 for the following trees that are also used in the Compliance Check application:
    - **hipaa_x12_4010.mtt**
    - **hipaa_x12_4010_type_2.mtt**
    - **hipaa_x12_4010a1_type_2.mtt**
14. Open the **compliance_check** map in Map Designer and choose **build all**. This will incorporate the newly changed and imported segment tree into c**ompliance_check**

Compliance check

Refer to the following procedures in order to verify compliance check testing using the newly created trees. There are two test files included in this package: **type_5_checks_valid.dat** and **type_5_checks_invalid.dat**.

The only difference is on line 23, the N402 element. The valid file has PA and the invalid file has PP:

To perform the compliance check perform the following steps:

1. Using the sample test data, run the **type_5_checks_valid.dat** through compliance_check.
2. Verify that the HIPAA_997 accepts all transactions.
3. Run the **compliance_check** again with **type_5_checks_invalid.dat**.
4. Check HIPAA_997 for AK3 and AK4 Errors. It will reject the transaction and specify as the error:

AK3*N4*21**8

AK4*2**7*PP

AK4*2**2

Type 5 implementation

The following paragraphs show how to implement Type 5 validation using a text file (no drivers) as well as how to implement Type 5 for component rules using database drivers.

Component rule using text file

To implement the Type 5 validation using a component rule instead of a restriction list, follow the steps in the previous section and adjust to use the executable map **add_type_5_component_rule** also located in **add_type_5_checks** map source.

Component rule using database drivers

You will find the following files and databases provided with the Type 5 example:

- **x12_external_code_source_22.txt** - text file used for both flat file and database text driver examples containing the valid state codes.
- **x12_external_code_source_22.mdb** - Microsoft Access database containing the valid state codes.
- **x12_external_code_source_22.xls** - Microsoft Excel spreadsheet containing the valid state codes.
- **add_type_5_checks.mms** - is the map source file. It contains the following executable maps:
  - **add_type_5_component_rule** - uses standard files as an input for the table.
  - **add_type_5_component_rule_mdb** - uses an access database table as input using the ODBC Access Driver.
  - **add_type_5_component_rule_xls** - uses an excel spreadsheet as input using the ODBC Excel Driver.

  - **add_type_5_component_rule_txt** - uses a text file as input using the ODBC Text Driver.
  - **add_type_5_restriction_list** - uses standard files as input for the table and puts the codes in a restriction list instead of a component rule.
- **x12_external_code_source.mtt** - type tree that provides generic type tree definitions of the state code table used in all maps.

The example executable map add_type_5_component_rule that doesn't use the database driver adds component rules by bringing all the values into the component rule and putting them in a member list. During compliance check, if the state cannot be found in the member list, an error is produced.

The sample executable maps that use the database drivers (**add_type_5_component_rule_txt**, **add_type_5_component_rule_mdb** and **add_type_5_component_rule_xls**) all add component rules which will do a DBLOOKUP at run time when the type tree is used in compliance check to determine if the code is valid. See the Database Interface Designer documentation for additional information.

Add and configure the ODBC Database drivers (MS Access, MS Excel and MS Text) in the ODBC Data Source Administrator using the same database names as provided in the IBM Database Query File with this example (**x12_external_code_source.mdq**). The default databases names are:
- **X12_External_Code_Source_txt**
- **X12_External_Code_Source_xls**
- **X12_External_Code_Source_mdb**

In the **add_type_5_checks** map, for the map executables that use the database, the second input card is only used to check connectivity with the applicable database. If the ODBC Driver is not set up correctly, you will not be able to connect and the map will fail.

When using the Excel spreadsheet, all entries must be selected and a name range created in order to correctly access the database. The spreadsheet provided (**x12_external_code_source.xls**) already has the name range defined as **code_source_22**.

When configuring the Microsoft Text Driver for the delimited text file, select **Options** -> **Define format**, highlight the file which contains the table data, check **Column name header**, select **CSV Delimited** and then choose **Guess Columns**.

Changing the ODBC data source name

If you want to change the names of the data sources, you need to change Input Card #2 to point to the new data source name as well as the rule that generates the validation in the output tree.

To change the ODBC Data Source name perform the following steps:
1. Open **f_Add_Component_Rule_22**.
2. View Output Card #1->Updated MTS-> Sequence Component #CHILDDATA->Rule Element->Rule #DATA
3. Search for DBLOOKUP. Replace the data source from the example with the actual data source name. See the DBLOOKUP function for more information.

Returning type 5 validation error codes

It should first be noted, that Type 5 checking is always on when Type 2 checking is on. See the Compliance Checking documentation for additional information.

When returning error codes from Type 5 validation, component rules give greater flexibility than restriction lists, however, restriction lists are more straightforward and operate faster.

The error returned from the restriction list will always be initiated with Type 2 checking, and recorded where Type 2 errors are recorded. The error code associated with a restriction list will always be AK403 = 7.

The data element number will always be AK401-1 = element count, upon which the restriction list is applied. When using component rules there is greater flexibility.

When using component rules, this current version of Type 5 validation has the capability to return errors on different elements or sub elements than the element that the component rule is located on, i.e. AK401-1 and AK401-2. This is required, as sometimes the component rule must be placed on a different element than it is reported on. This current version of Type 5 validation also gives the ability to change the AK403 error.

This is accomplished by placing a series of codes that are activated ONERROR. The generalized format for the string is E*e*E*S*s*S*C*c*C*T*t*T where:

- The capital letters and asterisks are the delimiters:
  - E = Element Number
  - S = SubElement Number
  - C = Error Code
  - T = Type of Validation
- The lower case letters are the number:
  - By default e = the element count in sequence that the component rule is on
  - By default s = the element count in sequence that the component rule is on
  - By default c = 2
  - By default t = 2

For example, the string E*11*E*S*4*S*C*7*C*T*5*T is interpreted as:

- Return an error on element 11 - AK401-1 = 11
- Return an error on subelement 4 - AK401-2 = 4
- Return error code 7 - AK403 = 7
- This is a type 5 validation

These codes can be seen in the example map provided.

To view the codes

1. In Map Designer, open the example map source: **add_type_5_checks**.
2. Open the functional map: **f_Add_Component_List_22_CLM**
3. View Output Card #1->Updated MTS-> Sequence Component #CHILDDATA->Rule Element->Rule #DATA

Note: This rule contains the previous example. The previous string must be placed as the second parameter in the ONERROR function in the component rule if the defaults need to be overridden.

Type 5 validation best practices

It is recommended that when adding multiple Type 5 validations, such as State, ICD-9, etc., that a map be created based upon the example provided for each validation. These maps should be saved and used anytime new versions of the Pack for HIPAA EDI are received. This will allow easy integration of Type 5 validation specific to the environment with new versions.

The choice of when to place the validation in a restriction list vs. component rule vs. database table depends upon the environment and the code set. If the code set changes frequently, database tables should make validation easier as the trees do not have to change frequently.

However, there will be a performance penalty when validating against an external database or file as opposed to a restriction list.

The original **hipaa_x12_4010_segment** tree and **hipaa_x12_4010** tree should be backed up and kept safe. If performance problems are found when using updated Type 5 trees, the same benchmark should be run with the original trees to determine if the difference is in the Type 5 validation. In addition, the original trees allow for debugging of possible problems independent of any changes.

Chapter 8. Type 6 validation

Type 6 testing ensures that data segments that differ, based on certain healthcare services, are properly created and processed into claims data formats.

There are two series of checks for type 6, automatic and manual.

Automatic checks are pre-configured and are active when Type 6 is active. For example, suppose that the type of service is an auto accident. In a case such as this, there cannot be an onset of current symptoms date. There is no user maintenance required for automatic type 6 checks.

Manual checks are made when the check must validate against multiple values specific to the type of service. These values are located in the implementation guides or external code sets. The user must maintain these values, as these values may change or will have to be located by using external codes sets.

The rest of this section details the manual checks. Automatic checks are active when Type 6 is active.

Type 6 cross-reference file

A Type 6 cross-reference file must be maintained to match line of business codes with relevant HIPAA and external values. This cross-reference file, called **hipaa_x12_type_6_value.dat**, is located in the data directory:

install_dir\**packs\healthcare_v***n.n*\**hipaa\data**

The Pack for HIPAA EDI reads this file during compliance check and any relevant services that are active in the HIPAA transmission are checked against this file during type 6 processing. To ensure that Type 6 is active, see the Compliance Checking documentation.

To maintain this file, a record must be entered and maintained for each type of service in each transaction set/version combination.

To enter and/or maintain a record, perform the following steps:

1. Turn on Type 6 processing as described in the Compliance Checking documentation.
2. Enter or update the line(s) in the cross-reference file for the transaction set and GS08 value that has a supported type of service.
3. Enter the values specific to this type of service located in the HIPAA implementation guides or external code sets.

To find the specific checks and values, the HIPAA implementation guides and possibly external code sets must be referenced.

Cross-reference file format

The cross-reference file is a pipe delimited text file that is described in the **compliance_check.mtt** type tree:

© Copyright IBM Corp. 2006

This cross-reference file is called the **hipaa_x12_type_6_value.dat** file, and it is located in the data directory in the following directory:

install_dir **\packs\healthcare_v***n.n***\\hipaa\data**

Notice that fields 1 and 2 describe the GS08 value and transaction set to use for this check.

Also notice, that fields 3 and 4 name this type of service and its code. Currently, IBM supports the following services for Type 6. These services, along with their associated code are as follows:

- Pregnancy - PREG
- Accident - ACC
- Home Health - HH
- Private Duty Nursing - PDN
- Services by a Nurses Agency - NA
- Spinal Manipulation - SPM
- Ambulance - AMB
- Oxygen Therapy - OXY

The services listed above, and their cross-reference codes are the only valid values in fields 3 and 4 of the file.

Fields 5, 6 and 7 describe the specific segment, element, and sub element, respectively, for this check. For example, if this check is for the UM segment 3rd element (UM03), field 5 would be UM, field 6 would be 3 and field 7 would be empty. If this check is for the SV101-02 sub element, field 5 would be SV1, field 6 would be 1, and field 7 would be 2.

Field 8 is the value(s) that must be satisfied for this check to pass. If there are multiple values in this field, the values are delimited by an asterisk (*). For example, if the values that must be checked are 1 or 2, field 8 contains 1*2. These values may be found in the implementation guides or in external code sets referenced by the implementation guides.

Field 9 is an On/Off indicator. A 1 means this record is active. A 0 means this record is inactive, and field 10 is a comment.

Validation example

For this example, pages 141-149 of the 278 HIPAA Implementation Guide will be used. Please refer to these pages.

The UM segment, 3rd element in the 278 Request has a Type 6 check as noted by the implementation guide. The cross-reference file would include these records:

004010X094|27Q|Pregnancy|PREG|UM|03||65*68*69*82*83*84|1|page 142

004010X094A1|27Q|Pregnancy|PREG|UM|03||65*68*69*82*83*84|1|page 142

In this case, the values came from the implementation guide.

Type trees

There are no Type 6 checks in the trees in the Pack for HIPAA EDI. Only by running the **compliance_check** map with Type 6 configured in the **compliance_check_parameter.dat** file and properly coded in the cross-reference file, will Type 6 checks be performed.

Chapter 9. Type 7 validation

Type 7 is specific to the HIPAA requirements that are related to Medicare, Medicaid, Indian Health and other partners. These requirements may be found in the HIPAA Implementation Guides or in specific documents distributed by trading partners.

Validation process

During compliance check, if type 7 checks are active, the Application Receiver's Code (GS03) data is examined and compared to a cross-reference file to determine if this partner has specific checks that must be validated. The user must maintain this cross-reference file.

This cross reference file is called the **hipaa_x12_type7_value.dat** located in:

*install_dir***\packs\healthcare_v***n.n***\hipaa\data**

During Type 7 processing, this file is read during compliance check. Any matching Application Reveiver's Codes between this file and the transmission are found and specific checks for that partner are performed. See the Compliance Checking documentation for more information.

To activate Type 7 checking for a partner, perform the following steps:

1. Enter the partner's GS03 value in the cross-reference file.
2. Enter the type of partner.

   Note: Currently this value must be Medicare.
3. Enter any comments.

Type 7 Cross Reference File

The cross-reference file is a pipe delimited text file that is described in the **compliance_check.mtt** tree.

This cross-reference file is the **hipaa_x12_type_7_value.dat** located in:

install_dir **\packs\healthcare_v***n.n***\hipaa\data**

The file contains the following fields:

- Field 1 is the Applications Receiver's Code (GS03).
- Field 2 is the type of partner. Currently this field must be Medicare.
- Field 3 is a comment.

During compliance check, this file is used to activate Medicare specific checks for GS03 matches from the transmission to this file. Currently, only Medicare is supported for specific checks. In addition, all transactions in the functional group will be validated for Medicare specific validation.

Validation example

For example purposes, suppose that there are two trading partners and both require Medicare checks. These partners have CMS1 and CLEARHOUSE1 as their Application Receiver's Code (GS03). In this case, the lines in the Type 7 cross-reference file must include these records:

CMS1|Medicare|To CMS

CLEARHOUSE1|Medicare|To my clearing house

Type trees

There are no Type 7 checks in the trees in the IBM Pack for HIPAA EDI. Only by running the compliance_check map with type 7 configured in the **compliance_check_parameter.dat** file, and properly coded in the cross-reference file will Type 7 checks be performed.

Chapter 10. Trading Manager

Because HIPAA X12 4010 and 4010 Addenda transaction sets make use of certain additional segments and code values that are not found in standard X12 4010, the X12 data validation capabilities of Trading Manager must be extended to accommodate HIPAA data. Trading Manager customers can use the objects in the Pack for HIPAA EDI to add the necessary components required for HIPAA transaction data:

- Type 1 - EDI Standard Integrity Testing.
- Type 2 through Type 7 - HIPAA Implementation Guide Requirement Testing.

Importing the type qualifier and X12 structure files

The following two files must be imported into Trading Manager before any HIPAA tradelinks can be created:

- **hipaa_x12_4010_qualifier.dat**
- **hipaa_x12_4010_structure.dat**

These files are located in:

install_dir\**packs\\healthcare_v***n.n*\**hipaa\\data**

If you are using HIPAA Inbound Trade Links with the Packs for HIPAA EDI you must first perform the steps in this section before converting from a prior release.

Before you begin, make certain that the current version of Trading Manager and the current version of the Pack for HIPAA EDI are both installed.

To import the type qualifier and structure files

1. Open **Partner Manager**.
2. From the **Tools** menu select **Standards**.

   The Standards Maintenance window opens.
3. On the **X12 Standards** pull-down menu, select **HIPAA Type Qualifiers: Import**, then click the **Find File** button.

   The Select/Enter HIPAA Rules/Type Qualifiers to Load dialog box opens.
4. Navigate to the following directory:

   install_dir\**packs\\healthcare_v***n.n*\**hipaa\\data**
5. Select the **hipaa_x12_4010_qualifier.dat** file and click **Open**.

   The dialog box closes.
6. Click the **Begin Import** button in the Standards Maintenance window to start the import.

   When the import completes a dialog box will open with the message **Import successful**.
7. On the **X12 Standards** pull-down menu, select **X12 Structure Validation: Import**, then click the **Find File** button.

   The **Select/Enter Structure File to Load** dialog box opens.
8. Navigate to the following directory:

   install_dir\**packs\\healthcare_v***n.n*\**hipaa\\data**

9. Select the **hipaa_x12_4010_structure.dat** file and click **Open**.
10. Click the **Begin Import** button in the Standards Maintenance window to start the import.

    When the import completes a dialog box will open with the message **Import successful**.

Setting up trade links for HIPAA X12 data

Refer to the Trading Manager documentation for information about setting up Trade Links.

EDI wizard

After you build your trade link definitions using the HIPAA X12 versions, you must run the EDI Wizard to create the **x12mail.mtt** type tree for Message Manager.

Use the **hipaa_x12_4010_type_1.mtt** type tree when pointing to the source locations that contain the standard and addenda HIPAA X12 4010 transactions. Select this tree as the source tree within EDI Wizard.

Note: Select either one of these trees regardless of the type of validation selected in Trading Manager.

Validating HIPAA X12 data

To accommodate WEDI/SNIP Type 1 through Type 7 testing within Trading Manager, you must copy the **x4010hipaasegmentdataaudit** and **x12type6qualifier** executable maps and the **hipaa_x12_type_6_value.dat** file for HIPAA data validation from the Pack for HIPAA EDI into Trading Manager.

Once it has been deployed, these components will be available for dynamic execution within the Message Manager to handle HIPAA X12 transaction sets and provide error reporting and 997/824 generation.

To implement testing functionality in Trading Manager:

1. Start the Message Manager application.
2. Under **Msg_Mgr RunMaps**, select **Validation**.
3. Click on any map below **Validation**.

   All validation maps are displayed in the window on the right.
4. Click the **Add Source Map** icon from the **Floating Toolbox**, and then click in the system window.

   The Add Source Map window appears.
5. Select the map source file which is located as follows:
   install_dir\**packs\healthcare_v***n.n***\hipaa\ maps\compliance_check.mms**
6. For the executable map, enable the check box for **x4010hipaasegmentdataaudit** and click **OK**.

   The **x4010hipaasegmentdataaudit** map is generated and appears in the system window.
7. For the **x4010hipaasegmentdataaudit** map, access the execution settings and change the map path to the target server.

   For example:

8. Click the **Add Source Map** icon again from the **Floating Toolbox**, and then click in the system window.

   The Add Source Map dialog appears.
9. Select the map source file which is located as follows: *install_dir*\**packs\\healthcare_v***n.n*\**hipaa\\ maps\\compliance_check.mms**
10. For the executable map, enable the check box for **x12type6qualifier** and click **OK**.

    The **x12type6qualifier** map is generated and appears in the system window.
11. For the **x12type6qualifier** map, access the execution settings and change the map path to the target server.

    For example:
12. Select **Msg_Mgr** → **RunMaps**
13. Under System, select Deploy > Definitions and checkmark "Transfer Additional Files".
14. While highlighting "Transfer Additional Files" click on **Details**.
15. Copy file **hipaa_x12_type_6_value.dat** from *install_dir*\**packs\\ healthcare_v***n.n*\**hipaa\\data** and paste it into the Details window from step 14.
16. Update the target location to point to the **tmgr/mmgr/share** folder on the Trading Manager target server.

    For example:
17. If you have not done so already, select **hipaa_x12_4010_type_1.mtt** type trees as the source tree when running the EDI Wizard for merging into the **x12mail.mtt** type tree.
18. After using the EDI Wizard, build the **x12mail.mtt** type tree.
19. From the Message Manager application, deploy the **MessageManager** system: right-click on the system and select **Deploy** → **DeployMessageManager** from the context menu.
20. In the **List** view of the Navigator, under the **Msg_Mgr** system definition file, right-click **Message_Manager** and select **Deploy** → **Definitions**.
21. From the Define Deploy Scripts for MessageManager dialog, select **Generate and Transfer Launcher Control File** and click **Details**.
22. Choose the path and filename for the Launcher control file. The default file name is **Msg_Mgr.msl**.
23. Click **Save** and then close the Define Deploy Scripts for MessageManager dialog.
24. Again, right-click the **MessageManager** system and select **Deploy** → **DeployMessageManager** from the context menu.
25. Double-click the **RunMaps** system. Go to the **System** menu and select **Deploy** → **BuildRunMaps**.

Chapter 11. Compliance check performance tuning and debugging

Several techniques can be applied to improve the performance of the Healthcare objects - and the Compliance Check application in particular. Some of these techniques rely on standard IBM's performance tuning. Others require an analysis of site-specific and task-specific requirements, and the deployment of particular architectures that optimize performance for those particular requirements.

IBM WebSphere Transformation Extender tuning

Since the Packs for HIPAA EDI are based on standard IBM maps and type trees, standard performance-tuning techniques within IBM Transformation Extender should be examined.

General IBM WebSphere Transformation Extender checklist

Check the following in the IBM WebSphere Transformation Extender.

1. Ensure that optimum performance of the IBM Transformation Extender is achieved. Refer to the IBM Transformation Extender Performance documentation.
2. Ensure that the Map Profiler is not enabled in the Map Designer. See the Map Designer documentation for additional information.
3. Ensure that the Map Debugger is not enabled in the Map Designer. See the Map Designer documentation for additional information.

Map settings

Ensure that the following map settings, including RUN function parameters, are set as described in the following checklist. Except where otherwise noted, refer to the IBM Transformation Extender Performance documentation instructions.

1. Change **WorkSpace** from File to Memory. This is particularly effective for smaller input/output sizes.
2. Increase/decrease **PageSize** and **PageCount** values.
3. Disable all audit logs (except the data audit log from the segment validation map).
4. Verify that all **Trace** settings are disabled.
5. Use the Merging and Pruning Type Trees options (see ″Merging and Pruning Type Trees″) when possible. For example, in a data transformation map for professional claims transaction sets, remove all other functional group and transaction set definitions from the HIPAA type tree that will be used for the input card.

Compliance check performance checklist

The options described in the following checklist can be modified in the **compliance_check_parameter.dat** file. Except where otherwise referred to other sources, see the documentation for the ″Compliance Check External Parameter File″ for additional information.

1. **Validation_Method** - Set to **M** (Map). This allows the preliminary validation component to determine which transactions are valid or invalid. Only invalid transactions will, therefore, be passed through the entire Compliance Check application.
2. **HIPAA_Response_Ack** - Set to **P** (pre 999) or **A** (after 999). This ensures that only one set of acknowledgements is produced.
3. **Paging_Parameter** - Ensure that this setting is optimum for the size of the data. Refer to the IBM WebSphere Transformation Extender Performance documentation for additional information.
4. **Work_File_Command** - Ensure that this setting is optimum for the size of the data. Refer to the IBM WebSphere Transformation Extender Performance documentation for additional information.
5. **HIPAA_Type_x_Checks_Enabled** - (Where x = 1, 2, 3 etc.) Ensure that only those checks required by your organization are turned on.
6. **HIPAA_Tack_Enabled** - Set to **N**. This suppresses the translated acknowledgement report.
7. **Detail_Max** parameters - Reduce the number of errors produced on acknowledgements to reduce the overall processing time.
8. Enable Multi-threaded execution.

Merging and pruning type trees

You can merge and prune type trees. Always practice safe file management. Copy the source type tree to another name before you make any modifications.

For example, to prune the **HIPAA_X12_4010.mtt** to contain only the X098 standards, copy the **HIPAA_X12_4010.mtt** type tree to **HIPAA_X12_4010X098.mtt**. The following steps describe the pruning procedure.

To prune transaction sets from a type tree

1. Save the source type tree under a new name (for example, **HIPAA_X12_4010X098.mtt**).
2. In the source **HIPAA_X12_4010.mtt** type tree, locate the **ANSI Funct'lGroup Partner Inbound** type.
3. Delete the unnecessary transactions sets under **ANSI Funct'lGroup Partner Inbound**.
4. Delete the same unnecessary transaction sets under **ANSI Funct'lGroup Partner Outbound**.

To create a destination type tree

1. Create a new type tree.
2. Rename the root type **EDI**.

To merge from the pruned tree to the destination type tree

1. In the pruned source type tree (for example, **HIPAA_X12_4010X098.mtt**) locate the **Transmission** type.
2. Right-click the **Transmission** type and select **Merge** from the context menu.

   The Merge Type dialog box appears.
3. Click in the new type tree window.

   The **To tree** field in the Merge Type dialog box is populated with the destination type tree name.
4. Click **Merge** in the Merge Type dialog box.

The new type tree is created.

5. Copy the **%_Type_Tree_Information** type from the source type as a subtype of EDI (the root type) in the destination type tree.
6. Update the **%_Type_Tree_Information** to reflect the changes you made.
7. Analyze the new type tree.
8. Save the new type tree under an appropriate name.

Find and replace

Both the Type Designer and Map Designer have find-and-replace options. In the Type Designer, the find-and-replace option is specifically for types. The Map Designer find-and-replace option is more comprehensive.

The Map Designer has separate **Find** and **Replace** options, as well as a **More Find/Replace** option. Use the **Find** option to find text within object names in a card, within map rules in a card, or within a page in the Organizer window. Use the **Replace** option to replace text within object names in a card, within map rules in a card, or within a page in the Organizer window The More Find/Replace command operates on the entire contents of a map source file. For example, you can search through cards and replace filenames ending with **.txt** with **.dat**; or, you can replace all card types named **Record Set Data** with **Record Data**.

To access the Find-and-Replace option

1. From the Type Designer menu or Map Designer menu, select **Edit** → **Find**. From the Map Designer only, you can also access the **More Find/Replace** option from the **Edit** menu.
2. For help on finding and replacing, right-click on the dialog and click **Help**.

Refer to the Map Designer documentation for detailed information about using this feature.

Example of using more find/replace

In the Map Designer, you can search map rules for the RUN function in the request maps in the **NCPDP_V5_1_Request_Report.mms** map source file.

Running the request maps displays the error message Source not available. You can use the More Find/Replace command to locate the true input source for these maps by searching for the **RUN** function, which specifies a map name.

Note: To view the entire map rule, double-click it in the **Rule** column.

Debugging

The following sections are provided to assist you in debugging the Pack for HIPAA EDI.

Fail function aborted map

Versioning is included to the **hcsvu** exit in the Compliance Check application. In the functional map, **f_CC_Debug_n_Param**, a call to the exit is performed in order to ensure the compatibility with the current version of the Compliance Check application.

If the Compliance Check application fails with the message ″Fail function aborted map″, it may be due to one of the following:

- **hcsvu** exit not found in the **Base_Exit_Directory** (defaults to system library path).
- **hcsvu** exit found but it is an incompatible version.
- actual message found in **compliance_check.log** in the system library path.

Platform specific configuring for z/OS

A DD statement is required in the Job Control Language (JCL) for the 8 output cards in **x12initialcontrolsummary.mmc** which are set to **!Create**. By default they need to be as follows:

```
//* DUMMY OUTPUT CARDS FROM X12INITIALCONTROLSUMMARY RUN
MAP
//*
//SUMMARY1 DD DUMMY
//ENVELOPE DD DUMMY
//CONTENT1 DD DUMMY
//CONTENT2 DD DUMMY
//SUMMARY2 DD DUMMY
//VALID1 DD DUMMY
//VALID2 DD DUMMY
//INVALID DD DUMMY
//*
```

The other 6 output cards to **x12initialcontrolsummary** use the Sink adapter and do not need any DD statements.

If the names of any of the output cards that are set to **!Create** are changed, the DD name must also be changed in the JCL. The format is the first 8 characters of the name, so ensure that these are unique.

XML validation for z/OS

The following points describe XML validation on z/OS.

- When performing XML Validation on z/OS, on either Batch or USS, you may have to perform some manipulation to the input files before proper validation can occur.
- The example file included with the Pack for HIPAA EDI, **275_4050_ig_examples.dat**, contains XML in the BIN segment which is encoded in UTF-8 format.
- The EDI data needs to be transferred in ASCII format in order to convert it to EBCDIC format, but the XML data needs to be transferred in Binary format. The file then needs to be put back together in order to validate the XML.
- The schemas required to validate the XML can be downloaded from www.HL7.org.
- On z/OS Batch, if the schema locations in the XML are DDNAMEs, then a DD reference is required in the JCL for each schema.

XML validation

The schemas required to validate the XML in the BIN segment of the **275_4050_ig_examples.dat** file are not included in the Pack for HIPAA EDI. These schemas can be downloaded from: www.hl7.org. The schemas must be installed in the map execution directory. If the Compliance Check application is unable to find the schemas, the **compliance_check_824h.out** file will contain many error messages but the first message will indicate that the primary document was not found.

Utilities validation

A utility map called **utilities_validation_check.mms** is included and installed in the directory:

install_dir**\packs\healthcare_v***n.n***\hipaa\maps**

Qualifier file tests

This map can be used to test the integrity of the Qualifier files which are used in the Compliance Check application.

The report that is created will be found in the data directory and is called the **utilities_validation.out**. When it runs, it checks the following files for content integrity:

- **hipaa_x12_4010_qualifier.dat**
- **hipaa_x12_type_6_value.dat**
- **hipaa_x12_type_7_value.dat**

The map uses the compliance_check.mtt. The input cards used are:

- **compliance_check_parameter.dat**
- **hipaa_x12_4010_qualifier.dat**
- **hipaa_x12_type_6_value.dat**
- **hipaa_x12_type_7_value.dat**

  The map should be run when an update is made to one or more of these files. See the Compliance Check documentation for information on when these files may need to be updated.

The map will obtain these files from the data directory:

*install_dir***\packs\healthcare_v***n.n***\hipaa\data**

It will automatically perform the following tests:

- **hipaa_x12_4010_qualifier.dat** - This validation map checks the basic integrity of the rules found in this file. It looks for duplicates (that is, whole rules that are duplicated in their entirety, or have duplicated rule numbers). It also looks for any rules that failed basic validation and any rules with an Error Text greater than 60 characters. It also checks Type 6 rules that have Line Of Business codes for a reciprocal entry in the hipaa_type_6_value.dat file. Any rules that have a Line Of Business code, but no Type 6 value entry, will be listed as potentially invalid. The report will indicate the area of the rule is potentially incorrect.
- **hipaa_x12_type_6_value.dat** and **hipaa_x12_type_7_value.dat** - These data files will be checked for basic validation, and the validation map will inform you of any rules that fail this check.

At the end of each check will be a running total of records found, and totals of any records that have failed the validation steps.

Debugging using the control summary file

To perform Control Summary File Debugging:

1. Set output card #4 in **x12initialcontrolsummary.mmc** to **Create**.
2. Rebuild the map.

The control summary file will contain the results of the checks done to that point (control summary, envelope checking, pre-content checking and content checking) for all the types of validation selected in the **compliance_check_parameter.dat** file.

Content checking consists of the following types of checks for each level of validation for each transaction set:

- Ensures that all segments are defined within the transaction set.
- Validates input data against the transaction structure as defined in the **compliance_check_structure.dat** file.
- If there are no errors in either of the previous checks, then the following checks are performed:
  - The loop Id is added to each segment in preparation of passing the data to segment validation.
  - Input data is validated against the rules in the **compliance_check_qualifier.dat** file.
  - Each segment ini the input data is validated against the **hipaa_x12_4010_segment.mtt** type tree.
-

Errors that result from the above validations are reformatted in a standard message format.

Once debugging is complete, set output card #4 back to **!Create**, and rebuild the map. This will ensure optimum performance of the Compliance Check application.

If x12initialcontrolsummary.mmc does not execute successfully, then all the acknowledgement files created will contain the following mesage:

```
x12allackgenerate.mmc was not executed and no acknowledgements were created.
```

In this event, one of the following probably occurred:

- data failed input type tree validation
- one of the executable maps called from **x12initialcontrolsummary** failed
- **hcsvu** exit failure

In order to help determine the cause of the failure, turn on output card #1in the main **compliance_check.mms** to provide the contents of the control summary file after the execution of **x12initialcontrolsummary.mmc**.

- If **x12initialcontrolsummary.mmc** does not execute successfully, a log file is created in the maps directory which indicatesif one of the inputs was invalid.
- By default, if any executable map fails execution, a log file is created in the map directory with a unique name.
- The control summary file provides failure details.

Debugging parameter file changes

When modifying the **compliance_check_paramter.dat** file ensure that you perform the following in order for the changes to be recognized.

Verify that all parameters are set as expected. To do this, view the top section of the compliance check summary file. All of the parameters and the settings used in the Compliance Check application are listed in that section.

- Remove the ″; ″ (semi-colon) from the beginning of the line being changed.
- For the following parameters, ensure that a space is entered prior to the command:

  - Paging_Parameter
  - Audit_Command
  - Work_File_Command
- Make sure that the value entered is a valid option for the parameter
- If the 999H acknowledgment is to be produced, the following 2 parameters must be modified:
  - **HIPAA_Response_Ack** - set to either **A** (After) or **B** (Both)
  - **HIPAA_999_Enabled** - set to either **A** (Always) or **E** (Errors)

Note: Do not modify the following files or the Compliance Check application may not function as expected:

- **compliance_check_structure.dat**
- **compliance_check_qualifier.dat**
- **tack_codetable.dat**

Data debugging

The Compliance Check application produces a report which translates the 997 and 824 responses into a more readable format. This report is helpful for identifying the exact error messages.

In order to create this report you make the following changes to the **compliance_check_paramter.dat** file:

- **HIPAA_TAck_Enabled** - set to **Y** (Yes)
- **HIPAA_TAck_Accepted** - by default it is set to **N** and will only produce the report for transaction sets in error.

The following files are included with the Pack for HIPAA EDI and are required in order to properly view the compliance_check_Tack.html:

- **minus.gif**
- **plus.gif**
- **compliance_check_TAck.css**

Multi-threaded execution

The Compliance Check application map included with the Pack for HIPAA EDI is not configured for multi-threaded execution for the benefit of application development. However, you can manually configure the Compliance Check application for multithreaded execution if needed.

By default, the Compliance Check application map is configured to facilitate a data certification process by generating execution audit logs that reflect the executable map. The key to effective multi-threaded execution is eliminating or reducing contention. This means ensuring the following:

- Audit logs and work files are either turned off or uniquely named.
- The execution environment is tuned to avoid input/output resource contention.

Parameter file adjustments

The following adjustments should be made to the **compliance_check parameter.dat** file in order to multi-thread the Compliance Check application.

- Work File Command: **-WU** or **-WM**
- Audit Command: **-AEU**

Launcher settings and tuning

The following adjustments must be made to the Launcher settings when multi-threaded execution of **compliance_check.mmc** is required. See the Integration Flow Designer documentation for more information about Launcher settings.

- Set **MapAudit AuditLocation** → **FileName** to **Unique**.
- Set **WorkSpace WorkFilePrefix** to **Unique**.
- Set Input Card #2 **SourceEvent** to **ON** and include a wildcard in the **FilePath** specification.
- Include a wildcard in the **FilePath** specifications for Output Card #2 through #8.
- Tune the **Retry** parameters for Input Cards #1 and #3.
- Tune the **PageSize** and **PageCount** parameters.

Implementation

Once the Compliance Check application for multithreading execution has been configured, use the following steps for implementation:

1. Re-build all the compliance check, X12 envelope check, and X12 content check executable maps.
2. Re-generate the **.msl** file and, if necessary, re-deploy the systems.
3. Re-start the Event Agent/Server.

Chapter 12. Acknowledgement errors

These lists show Compliance Check acknowledgement error codes.

AK304 - segment level

Code	Definition
AK304 = 1	Unrecognized segment ID
AK304 = 2	Unexpected segment
AK304 = 3	Mandatory segment missing
AK304 = 4	Loop Occurs Over Maximum Times
AK304 = 5	Segment Exceeds Maximum Use
AK304 = 6	Segment Not in Defined Transaction Set
AK304 = 7	Segment Not in Proper Sequence
AK304 = 8	Segment Has Data Element Errors

IK304 error code

Code	Definition
IK304 = 1	Unrecognized Segment ID
IK304 = 2	Unexpected Segment
IK304 = 16	Implementation dependent seg missing
IK304 = 4	Loop occurs > max allowed
IK304 = 5	Segment exceeds max use
IK304 = 14	Implementation ″not used″ seg pres
IK304 = 7	Segment not in sequence
IK304 = 8	Segment has data element errs

© Copyright IBM Corp. 2006

IK304 = 17	Implementation loop occurs < min times
IK304 = 18	Implementation segment below min use
IK304 = 19	Impl'n dependent not used seg present

AK403 - element level

Code	Definition
AK403 = 1	Mandatory data element missing
AK403 = 2	Conditional required data element missing
AK403 = 3	Too many data elements
AK403 = 4	Data element too short
AK403 = 5	Data element too long
AK403 = 6	Invalid character in data element
AK403 = 7	Invalid code value
AK403 = 8	Invalid date
AK403 = 9	Invalid time
AK403 = 10	Exclusion condition violated
AK403 = 12	Too many repetitions
AK403 = 13	Too many components

IK403 error code

Code	Definition
IK403 = 19	Impl'n dependent data element missing
IK403 = 2	Conditional required data element missing`
IK403 = 13	Too many components
IK403 = 4	Data element too short

IK403 = 5
Data element too long

IK403 = 6
Invalid character in data element

IK403 = 16
Code value not used in implementation

IK403 = 8
Invalid date

IK403 = 9
Invalid time

IK403 = 10
Exclusion Condition Violated

IK403 = 12
Too many repetitions

IK403 = I10
Implementation ″not used″ data element present

IK403 = I11
Implementation too few repetitions

IK403 = 12
Implementation Pattern Failure Match

IK403 = 13
Implementation dependent ″not used″ element present

AK502 - transaction set level

Code	Definition
AK502 = 1	Transaction set not supported
AK502 = 2	Transaction set trailer missing
AK502 = 3	Transaction set control number in header and trailer do not match
AK502 = 4	Number of included segments does not match actual count
AK502 = 5	One or more segments in error
AK502 = 6	Missing or invalid transaction set identifier
AK502 = 7	Missing or invalid transaction set control number
AK502 = 8	Authentication key name unknown
AK502 = 9	Encryption key name unknown
AK502 = 10	Requested service (authentication or encryption) not available

AK502 = 11
Unknown security recipient

AK502 = 12
Incorrect message length (encryption only)

AK502 = 13
Message authentication code failed

AK502 = 15
Unknown security originator

AK502 = 16
Syntax error in decrytped text

AK502 = 17
Security not supplied

AK502 = 18
Transaction set not in functional group

AK502 = 19
Invalid transaction set implementation convention reference

AK502 = 23
Transaction set control number not unique within the functional group

AK502 = 24
S3E security end segment missing for S3Ssecurity start segment

AK502 = 25
S3S security start segment missing for S3E security end segment

AK502 = 26
S4E security end segment missing for S4S security start segment

AK502 = 27
S4S security start segment missing for S4E security end segment

AK905 - functional group level

Code Definition

AK905 = 1
Functional group not supported

AK905 = 2
Functional group version not supported

AK905 = 3
Envelope trailer missing

AK905 = 4
Group control number in the functional group header and trailer do not agree

AK905 = 5
Number of included transaction sets does not match actual count

AK905 = 6
Group control number violates syntax

AK905 = 10
Authentication key name unknown

AK905 = 11
Encryption key name unknown

AK905 = 12
Requested service (authentication or encryption) not available

AK905 = 13
Unknown security requirement

AK905 = 14
Unknown security originator

AK905 = 15
Syntax error in decrypted text

AK905 = 16
Security not supported

AK905 = 17
Incorrect message length (encryption only)

AK905 = 18
Message authentication code failed

AK905 = 23
S3E security end segment missing for S3S security start segment

AK905 = 24
S3S security start segment missing for S3E security end segment

AK905 = 25
S4E security end segment missing for S4S security start segment

AK905 = 26
S4S security start segment missing for S4E security end segment

TA105 - interchange level

Code	Definition
TA105 = 001	The interchange control number in the header and trailer do not match
TA105 = 002	The standard as noted in the control standards identifier is not supported
TA105 = 003	This version of the controls is not supported
TA105 = 004	The segment terminator is invalid
TA105 =005	Invalid interchange ID qualifier for sender
TA105 = 006	Invalid interchange sender ID
TA105 = 007	Invalid interchange ID qualifier for receiver
TA105 = 008	Invalid interchange receiver ID
TA105 = 009	Unknown interchange receiver ID

TA105 = 010
Invalid authorization information qualifier value

TA105 = 011
Invalid authorization information value

TA105 = 012
Invalid security information qualifier value

TA105 = 013
Invalid security information value

TA105 = 014
Invalid interchange date value

TA105 = 015
Invalid interchange time value

TA105 = 016
Invalid interchange standards identifier value

TA105 = 017
Invalid interchange version ID value

TA105 = 018
Invalid interchange control number value

TA105 = 019
Invalid acknowledgement requrested value

TA105 = 020
Invalid test indicator value

TA105 = 021
Invalid number of included groups value

TA105 = 022
Invalid control structure

TA105 = 023
Improper (premature) end-of-file (transmission)

TA105 = 024
Invalid interchange content (e.g. invalid GS segment)

TA105 = 025
Duplicate interchange control number

TA105 = 026
Invalid data element separator

TA105 = 027
Invalid component element separator

TA105 = 028
Invalid delivery date in deferred delivery request

TA105 = 029
Invalid delivery time in deferred delivery request

TA105 = 030
Invalid deliver time code in deferred delivery request

TA105 = 031
Invalid grade of service code

TED01 - error reporting/acknowledgement process

Code	Definition
TED01 = 010	Total out of balance
TED01 = 848	Incorrect data
TED01-OTH	Used for reporting XML errors.
TED01-024	Other unlisted reason.

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

© Copyright IBM Corp. 2006

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

AIX
AIX 5L
AS/400
Ascential
Ascential DataStage
Ascential Enterprise Integration Suite
Ascential QualityStage
Ascential RTI
Ascential Software
Ascential
CICS
DataStage
DB2
DB2 Universal Database
developerWorks
Footprint
Hiperspace
IBM
the IBM logo
ibm.com
IMS
Informix
Lotus
Lotus Notes
MQSeries
MVS
OS/390
OS/400
Passport Advantage
Redbooks
RISC System/6000
Roma
S/390
System z
Trading Partner
Tivoli

WebSphere
z/Architecture
z/OS
zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

This product includes software developed by the Eclipse Project (http://www.eclipse.org/).

IBM WebSphere Transformation Extender Pack for HIPAA EDI, Version 4.2

Index

Printed in USA