HIPAA 5010 DDL Schemas
User Guide

October 2014
Trademarks

All terms mentioned in this guide that are known to be trademarks or service marks have been appropriately noted. WPC cannot attest to the accuracy of this information. Use of a term in this guide should not be regarded as affecting the validity of any trademark or service mark. “BizTalk” and “Microsoft” are registered trademarks of Microsoft Corporation.
Contents

1. Using the HIPAA 5010 DDL Schemas ........................................................................... 4
   1.1. 5010 DDL Schemas ................................................................................................. 4
   1.2. Execute a DDL script for a transaction against the target database server for as many transaction sets as needed. ................................................................................. 4
   1.3. Calling the DDL Schemas’ stored procedures for data insertion. ................................................. 5
   1.4. Customizing the DDL Schemas’ to remove schema binding. ................................................. 5

2. Data Model ....................................................................................................................... 5
   2.1. Data Model Description ............................................................................................. 5
   2.2. Base Tables .............................................................................................................. 6
   2.3. Views ....................................................................................................................... 7
   2.4. Querying the Database ............................................................................................. 8
1. Using the HIPAA 5010 DDL Schemas

1.1. 5010 DDL Schemas

The 5010 Data Definition Language (DDL) Schemas are SQL scripts that WPC has generated using the Microsoft 2013 BizTalk Server’s HIPAA 5010 Errata XSD schemas.

WPC has DDL Schemas for all 11 of the HIPAA-mandated ASC X12 5010 transactions, and 3 additional transactions:

- 270 – Health Care Eligibility Benefit Request
- 271 – Health Care Eligibility Benefit Response
- 276 – Health Care Claim Status Request
- 277 – Health Care Claim Status Response
- 277CA – Claim Acknowledgment (not HIPAA-mandated)
- 278 – Health Care Claim Services Review Request & Response
- 820 – Payroll Deducted and Other Group Premium Payment for Insurance Products
- 834 – Benefit Enrollment and Maintenance
- 835 – Health Care Claim Payment/Advice
- 837D – Health Care Claim: Dental
- 837I – Health Care Claim: Institutional
- 837P – Health Care Claim: Professional
- 997 – Functional Acknowledgment (not HIPAA-mandated)
- 999 – Implementation Acknowledgment (not HIPAA-mandated)

Each DDL Schema will create a SQL Server database with tables for all of the transaction loops, segments, and fields that are in the ASC X12 5010 transaction. The DDL Schemas include both used and not used segments and fields in order to provide flexibility to users.

1.2. Execute a DDL script for a transaction against the target database server for as many transaction sets as needed.

For each of the transactions to be stored, execute the corresponding script on the empty database. Here is an example using the “sqlcmd” utility:

```
& sqlcmd -S . -d 270 -i WPC.OnlyConnect.DBToolkit.270Database.sql -E -l
```

This command will execute the script to create database objects for a 270 transaction on the local SQL server in a database named [270] using Windows authentication, enabling quoted identifiers.
1.3. **Calling the DDL Schemas’ stored procedures for data insertion.**

The DDL Schemas include the following stored procedures that call the SQLCLR code to insert an X12 transaction in XML message format into the transaction database:

- `sp_LoadBizTalkXML`
- `sp_LoadBizTalkXML2`

Use `sp_LoadBizTalkXML2` when the XML message includes the ISA and GS envelope headers. Use `sp_LoadBizTalkXML` when the XML message does not include those envelope headers.

1.4. **Customizing the DDL Schemas’ to remove schema binding.**

By default, the DDL Schemas bind the created SQL database to the ASC X12 transaction’s XSD schema for data validation purposes. In some cases, users may not want that schema binding and validation. To create a database without the schema binding and validation, edit the following line:

```
[transactionDocument] [xml](CONTENT [dbo].[BTSschema]) NOT NULL,
```

To read as follows:

```
[transactionDocument] [xml] NOT NULL,
```

Be aware that removing the schema binding can result in the insertion of bad data.

2. **Data Model**

2.1. **Data Model Description**

The WPC ASC X12 Transaction SQL Server databases model the X12 technical report three (TR3) for a particular transaction set. These are commonly referred to as implementation guides. There is a common pattern for implementation between transactions sets that follows the TR3. Base tables are optimized for insertion as opposed to retrieval. Views of the data are built so that the names of segments and loops are easily found in the TR3 for the transaction. Consumers of the information are expected to read from the views. The views closely match the TR3. The goal is that a business analyst should not need to learn new vocabulary or semantics above what is in the technical reports. If you understand the TR3 you will know which view contains the data.

At the highest level there is a database for each transaction. The names of the databases are flexible. A common approach is to select the name of the database to match the number of the
transaction. For example, the 270 transactions would be stored in the [270] database. These databases are created as part of the setup. The WPC DDLs don’t assume a name.

2.2. Base Tables

The base tables collect data from the XML form of the transformed X12 EDI data, referred to as xEDI, as follows. Each database will have some common tables.

**EDITransaction**: Identifies the transaction. Any data in the Database ToolKit can be traced back to this table. This table has an identifier GUID, the date it was inserted in the Database ToolKit, and the control number of the transaction. If the ISA and GS segments are included in the transaction they are copied directly to columns in this table, as well as parsed in other tables.

**XMLDocument**: References the EDITransaction and stores the xEDI document from BizTalk. The data in the transactionDocument column is parsed out into other tables in the database. This offers choices for long-term storage. If other tables are truncated they can be re-populated with the original document. Also this column’s data could be removed and the information found in all the other tables in the database.

**ISA_ControlInterchangeHeader**: Contains the fields of the ISA segment for the referenced transaction. The 16 fields of the segment are kept in columns named ISA01 – ISA16. Explanation of the contents of each field can be found in Appendix B of the corresponding TR3.

**GS_FunctionalGroupHeader**: Contains the fields of the GS segment for the referenced transaction, with fields held in columns GS01 – GS08. Explanation of the contents of each field can be found in Appendix B of the corresponding TR3.

**Loops**: A transaction has a list of loops. A loop may have loops within it. This table references the transaction where the loop appears and does a self-join in the case where it is a loop within a loop. A loop record will have the loop name as found in the implementation guide. “STANDALONE” marks loops that are top level. This table is referenced by the rest of the tables in the database that match segments in a transaction and contain the data elements of those segments. Details of the references that are allowed between segment and loops, and loops to each other, can be found in the TR3 for the transaction.

Each transaction database contains these base tables that capture the basic structure of a transaction along with a set of tables mapping to segments in the technical report. The [270] has tables, one for each segment in the implementation guide, ST, BHT, HL, NM1, REF, N3, and so on. Each of these tables has columns named for the fields of the TR3. The NM1 table in the [270] database has columns named NM101 – NM111.
The base tables are not grouped with the same logic as the transactions. In the 270, for instance, the NM1 table holds segments that are from loops 2100A, 2100B, 2100C, and 2100D. To understand which NM1 the data represents requires looking up the loop record to which the NM1 refers and reading the loop name. Another thing to notice is that the predominant data type of the columns is VARCHAR. This is because the base tables are designed for collecting data efficiently from the source XML document. Analysis of the data is better performed in the views.

2.3. Views

The views in the databases map to loops in the TR3. In the 270 there is a 2000A loop holding Information Source Level information consisting of an HL segment. The [270] database has a TS270_2000A view with the information in that loop. It has columns referencing its loop and transaction (loopid, parentid, transactionid, ordinal), and it has data columns:

- HLid
- HL01__HierarchicalIDNumber
- HL03__HierarchicalLevelCode
- HL04__HierarchicalChildCode

The column names of the views are very close to the titles of the data elements found for this loop. Only the HL segments for this loop are in this view.

The STANDALONE view contains referential information plus data columns named:

- BHTid
- BHT01__HierarchicalStructureCode
- BHT02__TransactionSetPurposeCode
- BHT03__OriginatorApplicationTransactionIdentifier
- BHT04__TransactionSetCreationDate
- BHT05__TransactionSetCreationTime
- BHT06__TransactionTypeCode

Note that for this view, date and time fields are converted to DATETIME providing expected behavior, even though the actual data is stored as VARCHAR. In other words, the views have type information that is not present in the base tables.

The 2100B loop view demonstrates that information from more than one segment can be in a view. This view contains the columns of NM1, N3, N4, PER, and PRV segments, decorated with the names in the technical report.
The 2100B loop has repeating PE segments. When there are repeating segments in a transaction the Database ToolKit provides a view for the repeating segments referencing the loop where they are found. TS270_2100B_Per_InformationReceiverContactInformation can be seen as a “detail” for the TS270_2100B view to capture this situation.

Each database has an X12_EnvelopeHeader table capturing the original document as well as the ISA, GS, and ST segments.

Business analysts should be able to locate information from any transaction by selecting information from the views if they have a good understanding of the transaction’s TR3.

2.4. Querying the Database

When writing queries against the Database Toolkit views, the X12 TR3 for the transaction set should be used as the data dictionary for view navigation.

These are the basic rules for how the views work:

- There is one view created for each Loop referenced in the TR3. That view will contain all of the segments and elements that are marked in the XSD as “Used” for a loop and that do not repeat. Examples of these views are TS837_2000A and TS837_2010AA.
- There is one view created for each Segment in a loop that repeats. Again, it will only include elements marked “Used” in the XSD. An example of this type of view is TS837_2000A_NM1_BillingProviderName.
- In 5010 and later versions, composites are broken out into their own views. Views are only generated for the composites marked as “Used” in the XSD schema.
- Structural elements in the XSD that don’t match the structure in the TR3 are not included in the views. For example additional nodes for grouping or subloops would fit this category. So you wouldn’t find a view for instance named “NM1_Subloop_2”.

It is possible to use the relational data to query for a particular document and then retrieve the full XML that is in the XMLDocument table. That method is useful if you need only one transaction set and most of its data.

It is possible to use the views to “traverse” the various loops, i.e. 2000A down to 2010AA, etc.

- When traversing from a loop view to its parent, the key you want is: parentView.loopID = childView.parentID
- When traversing from a repeating segment view to the associated loop view you want: `segmentView.loopID = loopView.loopID`.
- When traversing from composite view (5010 and above) you want: `segmentView.id = compositeView.parentId` or `loopView.segID = compositeView.parentId`

The 5010 837 transactions have a unique situation with their 2300 loop in that it (and its children) can appear in two places in the hierarchy. It can have either the 2000B or the 2000C as parents. Any specific 2300 loop can only have one direct parent. The parent is either a specific 2000B or 2000C loop. In all cases, the 2000B or 2000C loop identifies the patient for the claim in the 2300 loop. Now, when the 2300 loop is ‘inside’ the 2000C loop, the patient is a dependent of a subscriber that is identified in a 2000B loop. So, the 2000C loop does have the 2000B loop as its parent. So, when a 2000C loop is the parent of a 2300 loop, the related 2000B loop is the grandparent of the 2300 loop.

The two basic structures are:

```
2000A
  2000B
    2300
```

OR

```
2000A
  2000B
    2000C
     2300
```

If there is a subscriber, Bob, with a claim and a dependent of that subscriber, Cathy, also with a claim then the 2000B loop must be repeated. The structure would be:

```
2000A
  2000B (HL04=0 – Bob)
    2300 (claim for Bob)
  2000B (HL04=1 – Bob)
    2000C (Cathy)
```
Consider the following queries:

```sql
SELECT TS837Q1_2300.* FROM TS837Q1_2300 INNER JOIN Loops ON TS837Q1_2300.parentid = Loops.id
WHERE (Loops.loopName = '2000B')
```

Is going to give you different results than:

```sql
SELECT TS837Q1_2300.* FROM TS837Q1_2300 INNER JOIN Loops ON TS837Q1_2300.parentid = Loops.id
WHERE (Loops.loopName = '2000C')
```

Composites have different identifiers depending on the parent of the 2300. This is reflected in the XSD. `dbo.TS837_2400_SV1_C003_CompositeMedicalProcedureIdentifier` only returns the 2400:SV101 data from 2300s that are direct children from the 2000B loop.

dbo.TS837_2400_SV1_C003_CompositeMedicalProcedureIdentifier_4 will contain the values for 2400:SV101 from the 2300s that are direct children of the 2000C loop. If you need both identifiers, you will need to build a UNION.

This is an example of querying for Line Item data:

```sql
FROM
dbo.TS837_2300 vw_2300
inner join dbo.TS837_2400 vw_2400
on vw_2300.loopid = vw_2400.parentid
left outer join dbo.TS837_2400_SV1_C003_CompositeMedicalProcedureIdentifier vw_2400_SV1_C003
on vw_2400_SV1id = vw_2400_SV1_C003.parentid
left outer join dbo.TS837_2400_REF_LineItemControlNumber vw_2400_REF_LICN
```
on vw_2400.loopid = vw_2400_REF_LICN.loopid

This approach is used because the dbo.TS837_2400_REF_LinItemControlNumber is a Segment view that you want to join its Loop view, so the instances will have the same loopid, i.e. it’s the same instance of the 2400 loop as the other values in the view.