FFE QHP Templates Preparation

This document describes the following:

A. Current process for filling out QHP templates
B. Available reference documents from CMS
C. Options for automating import of contents into QHP templates
D. Detailed guidelines for automating import of content for QHP templates

A. Current Steps for populating the QHP templates involve the following:

1. Issuers download the FFE Plan Management template (from FFE/HIOS application)
2. Issuers populate the Excel based template
3. Utilizing the built-in macro, Issuers validate and converts the template content to XML
4. Converted XML gets uploaded to FFE Plan Management

Above steps represent our intended design to provide QHP templates with built-in data validation and integrity checks. Without this built-in function, Issuers would have to conduct this function on their own.

- Given very short-time frame between finalization of QHP rules and starting of the QHP submission window, off-loading this data/integrity checks to Issuers would have been a tall order for Issuers to meet.

- By providing this built-in function, CMS is ensuring standardization of data being submitted by Issuers have met format, integrity, and validation thresholds. Given short-time frame to review & approve the submitted plans, without this functionality, it would likely lead to longer evaluation period (which CMS does not have).

Unintended consequence of built-in function is that it will make it more difficult to populate the templates in more automated fashion.

B. CMS will make the following reference documents available to Issuers:

1. FFE Plan Management templates
2. XMLs associated with each Plan Management templates
3. DTMs for each XMLs
4. FFE data dictionary for Plan Management and Unified Rate Review (does not contain any direct mapping of data dictionary elements to that of the FFE PM template elements)
5. Options for copy/paste automating import of contents into QHP templates
6. Sample JAVA codes to convert XML to Excel content
7. QHP PM User Guides

C. Options for importing contents into QHP templates

> Note: Direct upload via XML is not an available function for Year 1. CMS will evaluate this option for future years.

Option #1: All QHP PM Templates can be populated programmatically. Details are provided in the following pages (See Section D).

Option #2: Copy/pasting is permitted in all the Plan Management templates required as part of the Qualified Health Plan (QHP) submission. Helpful hints and details for pasting data into macro-enabled templates are provided in the CMS document titled 'Plan Management (PM) Copy & Paste in Templates'.

Option #3: Use program file coded in the Java source language to create Excel content from XML representation

> *CMS will provide Java code for each of the QHP templates that could be utilized by developers to create an Excel from another data source (i.e. XML representation of the QHP templates). We have included four Java code files that show sample code for the Network template so that interested Issuers can utilize them to test our their process.*

With this option, the Issuers will:

- Use the Java code as a model to build code to populate, from Issuer systems, the blank templates downloaded from CMS.
- Validates the data using the validate button in the downloaded and populated templates.
- Use the finalize button to generate the XML for upload to HIOS.

D. Detailed guidelines for automating import of content for QHP templates

Recommended software environments: FFE recommends issuers use the Java language and the Apache POI libraries to create the spreadsheets. CMS have utilized Apache POI libraries version 3.7

Data Dictionaries and details: Detailed Data Traceability Matrices (DTMs) are available to assist with filling out the templates. These spreadsheets contain all the data elements and their types and allowed values and function as data dictionaries. These documents must be followed for a successful load of the templates. We recommend that issuers extend these spreadsheets with additional columns capturing the source of their data. Once these mappings are established the developers can build the code to load the data into the spreadsheets.

Template details: Most templates are relatively simple and consist of rows of information or information that is set in a well-defined layout. It is critical that the programmer not change the rows or columns of the data elements. The validation and load functions require that these follow certain patterns. Each template has a hidden sheet called "Names" that contains the named ranges that are used to populate the drop-downs and pop-ups of the template. This sheet should never be deleted or edited. When filling out the templates programmatically, it will not be necessary to use the pop-ups or drop-downs that are built in for manual entry. However it is critical that when programmatically filling out templates, you use the same allowable values identified in the DTMs to populate cells. This will reduce validation errors.

The Benefits Template, the Prescription Drug template and the Rate Review templates are the most complex. However, even these can be handled by developers. The key will be to review the data dictionaries carefully and to work with the templates and macros in the delivered templates to understand the workings. In the following section we provide some guidance on how to fill out these templates.

The below are the steps to be followed to fill the templates:

1. Identify template and analyze it to identify the required data elements.
2. Extend the DTM to include details about the external data model.
3. Identify data source to populate template and establish any needed connectivity and/or services to retrieve the data.
4. Start with the downloaded template from the interfaces and programmatically open and identify the sheet to be filled. Ensure that the name of the sheet is correct and that the sheet is visible. Understanding the sheet structure is critical to this step, especially for the Plan and Benefits templates.
5. Work through the model and populate the fields. It is not necessary to add any formatting in this code. Save the file to the disk.
6. When you open the file you should see all the data. Validate that the data is correct.

7. Manually execute validate and finalize buttons. Ensure that the spreadsheet passes all validations. It may be necessary to work through the programs built in order to make this work correctly.

Submit multiple sheets with in a template:

In some of the templates there is a need to generate data in multiple sheets and submit templates with multiple sheets. But in all templates users are able to see only one visible sheet. In that case developers need to clone the corresponding hidden sheet and fill the data in newly created sheets. The below table provides the hidden sheet details for each template.

Template Name	**Hidden sheet name**	**Purpose**
Plan Benefits	DefaultBP	This is default Benefits Package sheet. If there is a need to submit multiple benefits packages then it is necessary to clone this sheet and name it as "Benefits Package N". Here N is the sequence.
	DefaultCSV	This is default Cost Share Variances sheet. For each Benefits package there should be corresponding Cost Share Variance Sheet that should exist. It is necessary to always clone this sheet for each Benefits package and name it as "Cost Share Variances N" . Here N is the sequence number and it should match with Benefits Package number.
	Names	This sheet will have all reference data. Do not modify any data. Once work book is generated just add "true" in cell M1. This will be used to format the workbook after generated programmatically.
Rate Data	Master	This is a blank copy of the Rate Tables sheet that is cloned in order to create all necessary sheets for the template. The sheet should be named in a similar fashion to the default sheet – Rate Table (N). Here N is the sequence.