



Smarter Balanced Assessment Consortium: Fixed Form Packager User Guide

August 29, 2018



Overview of the Fixed Form Packager.....	3
Installing, Building, and Running the Fixed Form Packager	3
Installation Requirements.....	3
Acquiring the Source Code	3
Building and Running the FFP	3
Command Line Options	4
Command Line Example	5
Schema Validation.....	5
GitLab Integration.....	5
GitLab Account	6
GitLab URL.....	6
GitLab Groups	6
GitLab API Token	6
Constructing the Excel Workbook Input File	7
Sheet Layout.....	8
Package Sheet	9
Scoring Sheet	10
Tests Sheet	12
Segments Sheet.....	12
SegmentForms Sheet	13
Tools Sheet	16

Overview of the Fixed Form Packager

The Fixed Form Packager (FFP) is a command-line Java application that ingests a Microsoft Excel workbook in .XLSX format and generates a fixed form test package XML document.

Installing, Building, and Running the Fixed Form Packager

Installation Requirements

This project uses the following tools for building and installing the Fixed Form Packager:

- Java 1.8
- Maven 3

Acquiring the Source Code

The source code for the FFP and additional installation instructions can be found at https://github.com/SmarterApp/TDS_FixedFormPackager. There are two options for downloading the source code. The first option is to navigate to the FFP web page where there is a button to download the source code as a zip file as shown in Figure 1.

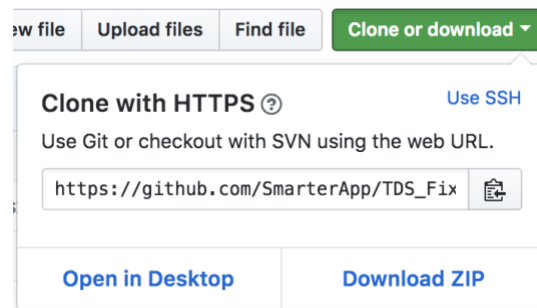


Figure 1: Download FFP code by clicking the "Clone or download" button.

Once the code has been downloaded, decompress the downloaded Zip file. This will result in the creation of the *TDS_FixedFormPackager* directory.

The second option for downloading the source code is to clone it using git with the shell command:

```
git clone https://github.com/SmarterApp/TDS\_FixedFormPackager.git
```

Both options will result in the creation of the *TDS_FixedFormPackager* directory.

Building and Running the FFP

After the installation requirements have been fulfilled, use a terminal application to execute the commands in this section to build and run the Fixed Form Packager.

Change directories to the *TDS_FixedFormPackager* directory and execute the following commands in a shell:

```
mvn clean install
```

This command will compile and run the tests for the Fixed Form Packager. As this command successfully completes, a new “target” directory will be created at the top of the TDS_FixedFormPackager directory. Inside this directory will be the newly built Java Archive file (JAR file) with a similar name to “target/tds-fixed-form-packager-081320182115.jar”. The last segment of the JAR’s filename is a timestamp indicating the date and time that it was built. Your filename will differ from the examples here, so be sure to use the correct file name.

```
java -jar target/tds-fixed-form-packager-081320182115.jar -h
```

This command will run the FFP and display the help message below detailing the command-line options available.

```
The fixed-form test package was not successfully created
Error message: Missing required options: t, g, u
usage: Sample usage: <INPUT XLSX> [OPTIONS]
-d,--debug          Prints more verbose debug output in case of errors
-g,--group <group>   *REQUIRED* GitLab Group
-h,--help           Prints help statement and exits
-o,--output <output> Output path of the generated test package file
                    (current directory by default)
-t,--token <token>   *REQUIRED* GitLab Token
-u,--url <url>       *REQUIRED* GitLab URL
```

Command Line Options

Along with the path to the Excel input file, the FFP has available a number of command line options, some of which are required, and some are optional. These command line options are detailed in Figure 2.

Argument	Long Option	Required	Description	Example
Path_to_XLSX	NA	Yes	The input Excel file containing the test package information. Must be in .XLSX format and not in .XLS format.	./SBAC-IAB-FIXED-G11M-Winter-2017-2018.xlsx
-g	--group	Yes	GitLab Group containing the Item metadata	-g ffp-development
-h	--help	No	Prints out the help/usage statement and exits.	-h
-o	--output	No	Output path of the	-o ffp-output-directory

			generated test package file. (current directory by default)	
-t	--token	Yes	GitLab token	-t YOUR_GITLAB_TOKEN
-u	--url	Yes	GitLab URL	-u https://gitlab-dev.smarterbalanced.org/
-d	--debug	No	Verbose logging which can help identify errors	-d

Figure 2: Command line options

Command Line Example

The example below leverages all of the command line options listed above.

```
java -jar target/tds-fixed-form-packager-081320182115.jar SBAC-IAB-FIXED-G11M-Winter-2017-2018.xlsx -g ffp-development -o ffp-output-directory -t YOUR_GITLAB_TOKEN -u https://gitlab-dev.smarterbalanced.org/ -d
```

The above command will take the Excel document `SBAC-IAB-FIXED-G11M-Winter-2017-2018.xlsx` as input and generate the test package file `ffp-output-directory/SBAC-IAB-FIXED-G11M.xml` assuming that the output directory `ffp-output-directory` previously exists.

Schema Validation

The FFP performs XSD schema validation on the test package XML document it produces. If the schema validation fails, however, it will still write the test package document to the output directory.

GitLab Integration

The Fixed Form Packager combines the information provided in the Excel input file with the information from the Item metadata.xml file and Item release data file (e.g. item-200-XXXX.xml) stored in a private GitLab instance to generate a fixed form test package. Where attributes from the Excel workbook and the GitLab information overlap, the Excel workbook takes precedence.


GitLab Account

Users of the FFP must have an account on the Smarter Balanced GitLab instance where the Item metadata is stored.

GitLab URL

Smarter Balanced operates its own instance of GitLab in order to store Item information. This is *not* the public instance of GitLab at <https://gitlab.com>. The FFP requires the URL of the GitLab instance that has the Item metadata and item release data XML files that will be included in the test package. This URL is specified with the `-u` or `--url` flag on the command line. An example of a URL might be <https://gitlab-dev.smarterbalanced.org/>.

GitLab Groups

You must be a member of the GitLab group you are attempting to access at least the Guest level. The group can be determined by browsing an Item's metadata and copying the URL segment between the base GitLab URL and the Item directory. For example, if you were browsing the Item metadata at <https://gitlab-dev.smarterbalanced.org/ffp-development/item-XXX-YYYYY> then the GitLab Group would be "ffp-development". You can find the groups you are a member of by clicking the  icon in the top-left corner of the GitLab webpage and selecting "Groups" from the drop down menu.

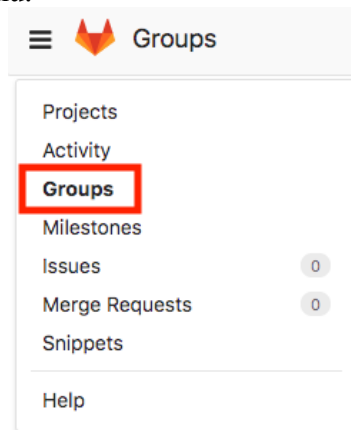


Figure 3: Select "Groups" from the dropdown menu.

GitLab API Token

Your GitLab token can be generated by going to your profile in your GitLab instance and selecting the "Access Tokens" link.

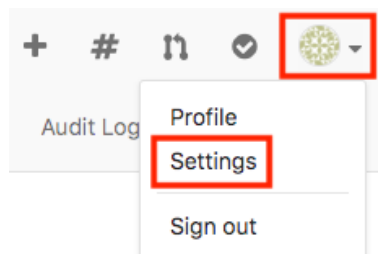


Figure 4: Click on the triangle in the top-right of the page and select "Profile"

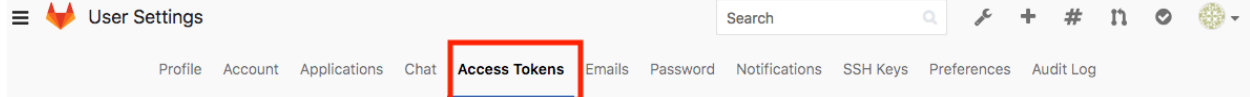


Figure 5: Click the link "Access Tokens."

If you do not already have a token, fill out the form as shown in Figure 6 to create one and copy the token value somewhere safe. This value is a *required* command line option (-t or --token) by the Fixed Form Packager in order for it to read the Item metadata and Item release files from the GitLab API.

1. Give the token a "Name" of your choosing.
2. Select the date for the token to expire.
3. Select the "api" and "read_user" checkboxes.

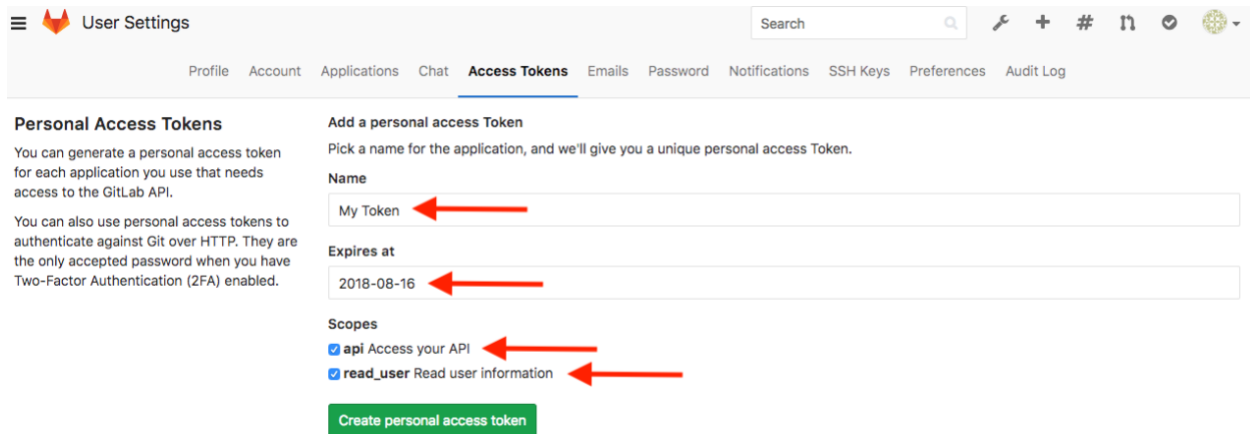


Figure 6: Fill out the token form as shown.

Once you fill out the form, you will be presented with your access token as shown in Figure 7. This is the first and only time your token will be presented to you. If you navigate away from this page, you will need to generate a new token. Copy your token value somewhere safe.

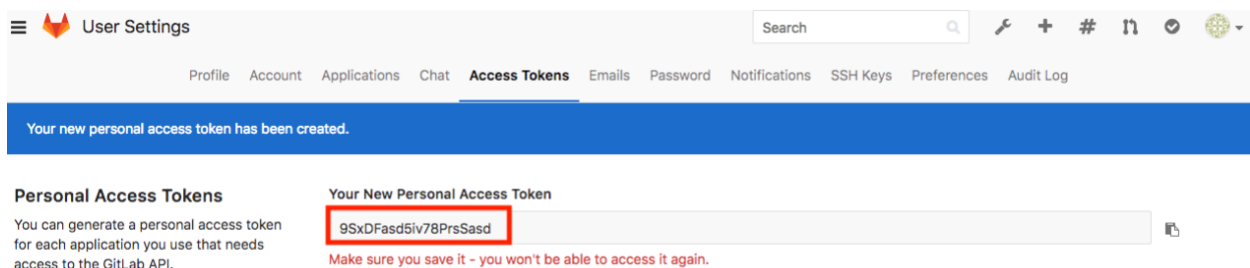






Figure 7: Copy your access token somewhere safe. Once you leave this page you won't be able to access it again.

Constructing the Excel Workbook Input File

The input document to the Fixed Form Packager is a Microsoft Excel workbook in .XLSX format (*not* .XLS format!). Examples can be found in the following table:

Test Package Name	Fixed Form Packager Input File
SBAC-IAB-FIXED-G7E-Winter-2017-2018	 SBAC-IAB-FIXED-G7 E-Winter-2017-2018.
SBAC-IAB-FIXED-G11M-Winter-2017-2018	 SBAC-IAB-FIXED-G1 1M-Winter-2017-201
SBAC-ICA-FIXED-G11E-Winter-2017-2018	 SBAC-ICA-FIXED-G1 1E-Winter-2017-201
SBAC-ICA-FIXED-G11M-Winter-2017-2018	 SBAC-ICA-FIXED-G1 1M-Winter-2017-201

Sheet Layout

The workbook contains multiple worksheets (Package, Scoring, Tests, Segments, SegmentForms, and Tools) as seen in Figure 8. You can select the different worksheets by clicking the tabs at the bottom of the Excel document.



Figure 8: The different worksheets as tabs on the bottom of the Excel workbook.

Each worksheet represents input to a section of the resulting test package document. For example, the highest level “Package” worksheet contains the highest-level information in the test package including the test package ID, the Bank Key, Academic Year, and more.

In addition, each worksheet has multiple header rows at the top describing the columns below and giving additional information to guide the entry of the data into the worksheet as shown in Figure 9.

Column Header	Description	Example						
Version	Version of the specification	2.0						
Input Data Values and Descriptions	Marks the columns that give descriptions of the input values as well as the variable name that the FFP uses to lookup the value	<table><tr><th colspan="2">Input Data Values and Descriptions</th></tr><tr><th>Input Variable Description</th><th>Input Variable</th></tr><tr><td>ID of the test package</td><td>Packageld</td></tr></table>	Input Data Values and Descriptions		Input Variable Description	Input Variable	ID of the test package	Packageld
Input Data Values and Descriptions								
Input Variable Description	Input Variable							
ID of the test package	Packageld							
# of Entries Allowed	Indicates whether the worksheet allows multiple columns of entries to be input for singular or repeating elements	Only 1 Package ID Allowed, 1 Or More TestId's Allowed Per TestPackageID, Multiple ItemId's Allowed Per Segment						

XML Reference	The hierarchical path to the element in the resulting XML document for the value in this row	TestPackage-publisher will fill the element attribute : <TestPackage publisher="SBAC" />								
Usage	Whether the row entry value is required or optional	<table><tr><td>Usage</td></tr><tr><td>Required</td></tr><tr><td>Required</td></tr><tr><td>Required</td></tr><tr><td>Optional</td></tr><tr><td>Optional</td></tr><tr><td>Optional</td></tr></table>	Usage	Required	Required	Required	Optional	Optional	Optional	
Usage										
Required										
Required										
Required										
Optional										
Optional										
Optional										
Input Variable Description	A helpful description of the value that should be entered in that row	ID of the test package, Test grade - comma separated list of grades								
Input Variable	Internal variable name that the FFP uses to look up the value	PackageId, SegmentFormPosition, ToolType_1								
Input Parameter	The value for that row. Enter your data in this colum.	<table><tr><td>Input Variable</td><td>Input Parameter</td></tr><tr><td>PackageId</td><td>SBAC-ICA-FIXED-G11E</td></tr><tr><td>BankKey</td><td>200</td></tr><tr><td>AcademicYear</td><td>Winter-2017-2018</td></tr></table>	Input Variable	Input Parameter	PackageId	SBAC-ICA-FIXED-G11E	BankKey	200	AcademicYear	Winter-2017-2018
Input Variable	Input Parameter									
PackageId	SBAC-ICA-FIXED-G11E									
BankKey	200									
AcademicYear	Winter-2017-2018									

Figure 9: Worksheet headers.

The areas of the worksheet are also color-coded to separate the template and description areas. The cells with both dark and light gray background are the template areas and generally should not have their values changed. The only exception to this is when repeating elements are added by copying and pasting rows at the bottom of the document and changing the Input Variable names to reflect the new set of parameters as with the ItemScoreParameters in the SegmentForms sheet. Additional description of this scenario will be discussed in the worksheets that require these changes. The user input area has a white background and is where the user should enter their data.

The next sections will discuss each of the worksheets, working from left to right across the tabs on the bottom of the Excel document starting with the Package worksheet and ending with the Tools worksheet.

Package Sheet



Figure 10: The Package worksheet.

The Package sheet defines the entries for the top-level <TestPackage> element as well as the Blueprint/Scoring/PerformanceLevels. All of the entries are required except for the AssessmentSubtype row.

Scoring Sheet

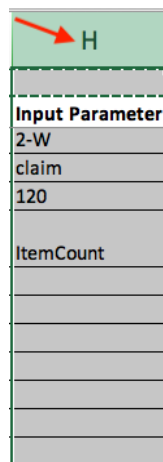


Figure 11: The Scoring worksheet.

The Scoring worksheet primarily defines the BlueprintElement/Scoring/Rules section of the test package. The first 4 rows, BlueprintElementId, BlueprintElementType, ComputationalOrder, and Rule name, are required, and the rest are optional.

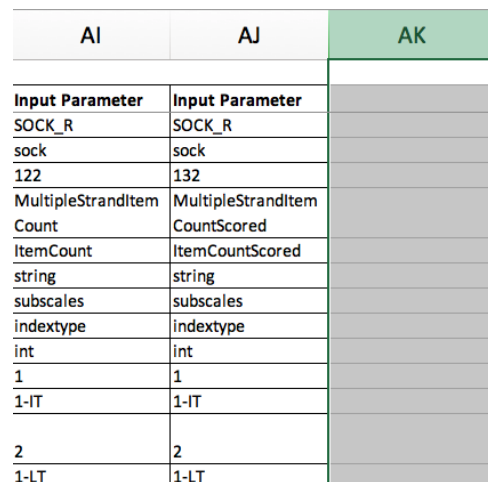
Adding Additional Rules

As indicated by the “One Rule Per Column” header, additional Rules can be added to the Scoring worksheet by adding additional columns to the right of the last column and keeping the headers for that column intact. A reliable technique for accomplishing this without error is to select an entire “Input Parameter” column by clicking on the lettered column indicator at the very top of the Excel worksheet (for example column “H”) to highlight the entire column (see **Error! Reference source not found.**), use your computer’s copy feature, and then enter it into the column after the last “Input Parameter” to the right side of the worksheet by again selecting the first empty column’s lettered column header (for example column “AK” in Figure 13) and using paste. Then change any fields that need to be changed in that column.



H
Input Parameter
2-W
claim
120
ItemCount

Figure 12: Select an entire row to copy by clicking on the column header.



AI	AJ	AK
Input Parameter	Input Parameter	
SOCK_R	SOCK_R	
sock	sock	
122	132	
MultipleStrandItem	MultipleStrandItem	
Count	CountScored	
ItemCount	ItemCountScored	
string	string	
subscales	subscales	
indextype	indextype	
int	int	
1	1	
1-IT	1-IT	
2	2	
1-LT	1-LT	

Figure 13: Select first empty column to paste copied column into.

Rule Parameters

Rule Parameters are defined in groups per Rule column by the naming convention of the Input Variable column. The rule parameters use a numbering convention of <PARAM>_X_Y where X and Y are integer values. The values used for X and Y must be sequential (e.g. 1, 2, 3, etc.).

If there is only one set of Rule Parameters defined in the input Excel document, then the user would fill out the values for the group containing optional Input Variables stopping before reaching ParameterType_2, which marks the beginning of the next group of Rule Parameters. This example is shown below:

```
ParameterType_1,
ParameterName_1,
ParameterPropertyName_1_1,
ParameterPropertyValue_1_1,
ParameterComputationRuleIndex_1_1,
ParameterComputationRuleValue_1_1,
ParameterComputationRuleIndex_1_2,
ParameterComputationRuleValue_1_2,
ParameterComputationRuleIndex_1_3,
ParameterComputationRuleValue_1_3
```

The first integer after the first underscore “_” indicates the Rule Parameter group. Within a group of Rule Parameters there *may* be more than one pair of ParameterComputationRuleIndex and ParameterComputationRuleValue entries. These are indicated by the second integer in the Input Variable name. For example ParameterComputationRuleIndex_1_2, and ParameterComputationRuleValue_1_2 are the second set of ParameterComputationalRuleIndex and ParameterComputationRuleValues within the first set of Rule Parameters.

To add an additional set of Rule Parameters, copy all of the rows from one of the existing sets and paste it at the bottom being careful to properly modify the two integer values after the underscores. For example, if a 5th Rule Parameter needed to be added, the user could copy the rows for the 4th group and paste them below the last row as seen in Figure 14.

38	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter-type	Optional	Parameter data type	ParameterType_4	string
39	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter-name	Optional	Parameter name	ParameterName_4	strands
40	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Property-name	Optional	Parameter Property name	ParameterPropertyName_4_1	indextype
41	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Property-value	Optional	Parameter property type	ParameterPropertyValue_4_1	int
42	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-index	Optional	Parameter computational rule index	ParameterComputationRuleIndex_4_1	1
43	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-value	Optional	Parameter computational rule value	ParameterComputationRuleValue_4_1	1-IT
44	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-index	Optional	Parameter computational rule index	ParameterComputationRuleIndex_4_2	2
45	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-value	Optional	Parameter computational rule value	ParameterComputationRuleValue_4_2	1-LT
46	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-index	Optional	Parameter computational rule index	ParameterComputationRuleIndex_4_3	
47	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-value	Optional	Parameter computational rule value	ParameterComputationRuleValue_4_3	
48	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter-type	Optional	Parameter data type	ParameterType_4	string
49	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter-name	Optional	Parameter name	ParameterName_4	strands
50	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Property-name	Optional	Parameter Property name	ParameterPropertyName_4_1	indextype
51	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Property-value	Optional	Parameter property type	ParameterPropertyValue_4_1	int
52	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-index	Optional	Parameter computational rule index	ParameterComputationRuleIndex_4_1	1
53	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-value	Optional	Parameter computational rule value	ParameterComputationRuleValue_4_1	1-IT
54	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-index	Optional	Parameter computational rule index	ParameterComputationRuleIndex_4_2	2
55	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-value	Optional	Parameter computational rule value	ParameterComputationRuleValue_4_2	1-LT
56	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-index	Optional	Parameter computational rule index	ParameterComputationRuleIndex_4_3	
57	TestPackage.Blueprint.BlueprintElement.Scoring.Rules.Rule.Parameter.Value-value	Optional	Parameter computational rule value	ParameterComputationRuleValue_4_3	

Figure 14: Copying and pasting a Rule Parameter group.

Then the user should clear out the Input Parameter values and renumber the Input Variable values to

```
ParameterType_5,
ParameterName_5,
ParameterPropertyName_5_1,
ParameterPropertyValue_5_1,
ParameterComputationRuleIndex_5_1,
ParameterComputationRuleValue_5_1,
ParameterComputationRuleIndex_5_2,
ParameterComputationRuleValue_5_2,
ParameterComputationRuleIndex_5_3, and
```

ParameterComputationRuleValue_5_3
as seen in Figure 15.

ParameterType_4	string
ParameterName_4	strands
ParameterPropertyName_4_1	indextype
ParameterPropertyValue_4_1	int
ParameterComputationRuleIndex_4_1	1
ParameterComputationRuleValue_4_1	1-IT
ParameterComputationRuleIndex_4_2	2
ParameterComputationRuleValue_4_2	1-LT
ParameterComputationRuleIndex_4_3	
ParameterComputationRuleValue_4_3	
ParameterType_5	
ParameterName_5	
ParameterPropertyName_5_1	
ParameterPropertyValue_5_1	
ParameterComputationRuleIndex_5_1	
ParameterComputationRuleValue_5_1	
ParameterComputationRuleIndex_5_2	
ParameterComputationRuleValue_5_2	
ParameterComputationRuleIndex_5_3	
ParameterComputationRuleValue_5_3	

Figure 15: Change Rule Parameter group numbers and enter new data.

Tests Sheet



Figure 16: The Tests worksheet.

The Tests worksheet identifies one or more <Test> elements in the test package. Both the TestId and TestLabel are required and must be unique. More than one Test can be defined by adding additional columns to the right on the worksheet as seen in Figure 17.

1 Or More TestId's Allowed Per TestPackageID		
Input Variable	Input Parameter	
TestId	SBAC-ICA-FIXED-G11E	SBAC-ICA-FIXED-G11E-Perf-HowWeLearn
TestLabel	High School ELA Interim Test (ICA)	High School ELA Performance Task - How We Learn (ICA)

Figure 17: Two Tests defined with two Input Parameter columns.



Reminder: The TestId values are referenced in both the Segments worksheet and the Tools worksheet. Make sure that the values are the same.

Segments Sheet



Figure 18: The Segments worksheet.

The Segments worksheet defines the one or more <Segment> elements within a <Test> element.

Segments are linked to a Test via the TestId in that the TestId entered in the Segments worksheet must match a Test from the Tests worksheet. If more than one Segment has the same TestId, then those segments will appear in the same Test, and their order will be determined by the SegmentPosition. For example, Figure 19 shows 3 different Segments with the following SegmentIds:

- SBAC-ICA-FIXED-G11E
 - Belongs to Test with TestId **SBAC-ICA-FIXED-G11E**
- SBAC-ICA-FIXED-G11E-Perf-HowWeLearn**A**
 - Belongs to Test with TestId **SBAC-ICA-FIXED-G11E-Perf-HowWeLearn**
- SBAC-ICA-FIXED-G11E-Perf-HowWeLearn**B**
 - **Also** belongs to Test with TestId **SBAC-ICA-FIXED-G11E-Perf-HowWeLearn**

Input Variable	Input Parameter	Input Parameter	Input Parameter
SegmentId	SBAC-ICA-FIXED-G11E	SBAC-ICA-FIXED-G11E-Perf-HowWeLearnA	SBAC-ICA-FIXED-G11E-Perf-HowWeLearnB
TestId	SBAC-ICA-FIXED-G11E	SBAC-ICA-FIXED-G11E-Perf-HowWeLearn	SBAC-ICA-FIXED-G11E-Perf-HowWeLearn
SegmentPosition	1	1	2
SegmentLabel	SBAC-ICA-FIXED-G11E	SBAC-ICA-FIXED-G11E-Perf-HowWeLearnA	SBAC-ICA-FIXED-G11E-Perf-HowWeLearnB

Figure 19: Three Segments belonging to two different tests.

The second and third Segments, SBAC-ICA-FIXED-G11E-Perf-HowWeLearnA and SBAC-ICA-FIXED-G11E-Perf-HowWeLearnB, are both members of the Test with TestId SBAC-ICA-FIXED-G11E-Perf-HowWeLearn. As such they must be ordered within the Test, which is accomplished by assigning the appropriate SegmentPosition to each of them.

To add additional Segments add additional columns to the right of the worksheet as discussed in previous sections.



Reminder: The SegmentId values are referenced in SegmentForms worksheet. Make sure that the values match where appropriate.

SegmentForms Sheet



Figure 20: The SegmentForms worksheet.

SegmentForms Overview

The SegmentForms worksheet defines the SegmentForms, Presentations, ItemGroups, and Items within a test package and associates them with the proper Segments. The input to this worksheet is a set of columns each with attributes about a specific item in a test package. Based on those attributes of an Item like SegmentFormId, SegmentFormPosition, and ItemPosition, as well as whether or not it has an Associated Stimulus Id (<associatedpassage>) in the GitLab item release metadata, determines the grouping of SegmentForms, ItemGroups, and Items. All items referenced on the SegmentForms sheet must be accessible in GitLab as discussed in the GitLab integration section.



Reminder: The SegmentId values referenced in SegmentForms worksheet must match with SegmentIds in the Segments worksheet.

Relationships and Rules

The following rules illustrate the relationships established by the Item Input Parameters:

- Items with the same SegmentId belong to the same Segment.
- Items with the same SegmentFormId belong to the same SegmentForm within the same Segment.
- Items with the same Associated Stimulus Id from the GitLab item metadata belong to the same ItemGroup and should be placed next to each other in adjacent columns. Note that the Associated Stimulus Id is not represented in this sheet *or* in the entire FFP input workbook.
- If an Item does not have an Associated Stimulus Id in GitLab, then it will be placed in its own ItemGroup with no other Items.

For example, in Figure 21 can be seen 2 different SegmentIds (SBAC-IAB-FIXED-G11M-AlgLinearFun-NoCalc and SBAC-IAB-FIXED-G11M-AlgLinearFun-Calc) representing 2 different Segments that the Items in the table will belong to with the first 3 Items belonging to the NoCalc Segment and the second 3 Items belonging to the Calc Segment. The SegmentFormIds for them can match the SegmentIds in this case because there is only one SegmentForm per Segment. This is also the reason that the SegmentFormPosition is “1” across all of the items; there is only one SegmentForm in each Segment, and thus it has to be the first SegmentForm in the Segment.

The first SegmentForm with the first 3 items in Figure 21 (SBAC-IAB-FIXED-G11M-AlgLinearFun-NoCalc) shows ItemPositions of “1”, “2”, and “3” respectively, ordering them within the SegmentForm. What determines the ItemGroup grouping is the Associated Stimulus Id (<associatedpassage>) in the GitLab item release metadata. If an item does not have one defined or it is blank, then the item will go into its own ItemGroup. If two items have the same Associated Stimulus Id, then they will go into the same ItemGroup and be ordered by their ItemPosition. Items with the same Associated Stimulus Id should be adjacent to each other in the SegmentForm.

Input Variable	Input Parameter	Input Parameter	Input Parameter	Input Parameter	Input Parameter	Input Parameter
SegmentId	SBAC-IAB-FIXED-G11M-AlgLinearFun-NoCalc	SBAC-IAB-FIXED-G11M-AlgLinearFun-NoCalc	SBAC-IAB-FIXED-G11M-AlgLinearFun-NoCalc	SBAC-IAB-FIXED-G11M-AlgLinearFun-Calc	SBAC-IAB-FIXED-G11M-AlgLinearFun-Calc	SBAC-IAB-FIXED-G11M-AlgLinearFun-Calc
SegmentFormId	SBAC-IAB-FIXED-G11M-AlgLinearFun-NoCalc	SBAC-IAB-FIXED-G11M-AlgLinearFun-NoCalc	SBAC-IAB-FIXED-G11M-AlgLinearFun-NoCalc	SBAC-IAB-FIXED-G11M-AlgLinearFun-Calc	SBAC-IAB-FIXED-G11M-AlgLinearFun-Calc	SBAC-IAB-FIXED-G11M-AlgLinearFun-Calc
SegmentFormCohort	Default	Default	Default	Default	Default	Default
SegmentFormPosition	1	1	1	1	1	1
ItemPosition	1	2	3	1	2	3
ItemId	83834	83838	12164	501	35334	33727

Figure 21: A set of Items from the SegmentForm sheet.

ItemScore Parameters

ItemScore Parameters are optional in the SegmentForms sheet. If none are found, they will be retrieved from the GitLab Item metadata’s <IrtDimension> elements. This is, however, an all or

nothing scenario in that if the first MeasurementModel_1 Input Variable is defined in the Excel sheet, the FFP will assume that all of the ItemScore Parameters will come from the Excel SegmentForms sheet and not look in GitLab for them. In addition, once a MeasurementModel_{N} is empty in the SegmentForms sheet for that Item, the FFP will stop processing ItemScore Parameters for that Item.

The FFP user can define any number of ItemScoreDimension Groups by adding additional rows to the bottom of the SegmentForms sheet with the additional Input Variables that define an ItemScore Parameter group and incrementing the integer after the underscore in the Input Variable name. The parameters are:

- MeasurementModel_N
- ScorePoints_N
- Dimension_N
- a_N
- b_N
- b0_N
- b1_N
- b2_N
- b3_N
- b4_N
- c_N

where N is the number of the ItemScoreDimension Group. The values used for N must be sequential (e.g. 1, 2, 3, etc.). If the SegmentForms worksheet has up to MeasurementModel_3 through c_3 defined, and another is needed, copy the last ItemScoreDimension Group's rows, and paste them below, being careful to increment the trailing integer for those new rows to MeasurementModel_4 through c_4 as seen in Figure 22 and Figure 23.

45	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension-measurementModel	Optional		MeasurementModel_3
46	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension-scorePoints	Optional		ScorePoints_3
47	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension-dimension	Optional		Dimension_3
48	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension-weight	Optional		Weight_3
49	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		a_3
50	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b_3
51	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b0_3
52	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b1_3
53	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b2_3
54	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b3_3
55	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b4_3
56	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		c_3
57	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension-measurementModel	Optional		MeasurementModel_4
58	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension-scorePoints	Optional		ScorePoints_4
59	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension-dimension	Optional		Dimension_4
60	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension-weight	Optional		Weight_4
61	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		a_4
62	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b_4
63	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b0_4
64	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b1_4
65	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b2_4
66	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b3_4
67	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		b4_4
68	TestPackage.Test.Segments.Segment.SegmentForms.SegmentForm.ItemGroup.Item.ItemScoreDimension.ItemScoreParameter-measu	Optional		c_4

Figure 22: Adding an additional ItemScoreDimension group.

MeasurementModel_3
ScorePoints_3
Dimension_3
Weight_3
a_3
b_3
b0_3
b1_3
b2_3
b3_3
b4_3
c_3
MeasurementModel_4
ScorePoints_4
Dimension_4
Weight_4
a_4
b_4
b0_4
b1_4
b2_4
b3_4
b4_4
c_4

Figure 23: Zoomed in view of the additional ItemScoreDimension group.

Tools Sheet



Figure 24: The Tools worksheet.

The Tools worksheet (Figure 24) defines the <Tools> and <Tool> entries which may appear within either the <Test> element or the <Segment> element. Tools are associated with either the Test or Segment element by the required entry TestOrSegmentId in the Tools worksheet that much match *either* a Test or SegmentId from the Tests or Segment worksheets. Tools groups can be defined as additional column entries in the Tools worksheet by adding new columns to the right of the worksheet. Individual Tool entries use a numbering convention of <TOOL>_X_Y where X and Y are integer values. The values used for X and Y must be sequential (e.g. 1, 2, 3, etc.). They are defined within a single column by repeating a Tool entry and incrementing the integer value starting with ToolName_N and ending with ToolOptionDependencyDefault_N_2_2 where N increases with the number of Tool entries within a Tools group. For example the third Tool entry within a Tools element would begin with ToolName_3 and end with ToolOptionDefault_3_2 as shown in Figure 25.

ToolName_3	American Sign Language
ToolType_3	
ToolStudentPackageFieldName_3	
ToolSortOrder_3	
ToolRequired_3	TRUE
ToolDisableOnGuest_3	FALSE
ToolAllowMultiple_3	FALSE
ToolAllowChange_3	TRUE
ToolOptionCode_3_1	TDS_ASLO
ToolOptionCode_3_2	TDS_AS1
ToolOptionSortOrder_3_1	0
ToolOptionSortOrder_3_2	1
ToolOptionLabel_3_1	Do not show ASL videos
ToolOptionLabel_3_2	Show ASL videos
ToolOptionDefault_3_1	TRUE
ToolOptionDefault_3_2	FALSE

Figure 25: Input Variables and values for a third Tool entry.



Reminder: The TestOrSegmentId values must match either a TestId or SegmentId from the Tests or Segments worksheet.

Tool Options

A tool may have a number of tool Options defined and several Tool Dependencies within the Options. Additional Tool Options are defined by repeating the ToolOptionCode_X_Y, ToolOptionSortOrder_X_Y, ToolOptionLabel_X_Y, and ToolOptionDefault_X_Y Input Variables within a specific Tool definition where “X” is the Tool number and “Y” is the Option number within that Tool’s Options group. For instance, Figure 25 demonstrates an example of a Tool definition that has two Tool Option entries, the first is defined by the following Input Variables:

- ToolOptionCode_3_1
- ToolOptionSortOrder_3_1
- ToolOptionLabel_3_1
- ToolOptionDefault_3_1

The second group is defined by the following Input Variables:

- ToolOptionCode_3_2
- ToolOptionSortOrder_3_2
- ToolOptionLabel_3_2
- ToolOptionDefault_3_2

The resulting XML elements is shown in Figure 26:

```
<Tool name="American Sign Language" allowChange="true" allowMultiple="false" sortOrder="3" disableOnGuest="false" required="true">
  <Options>
    <Option default="true" label="Do not show ASL videos" code="TDS_ASLO" sortOrder="0"/>
    <Option default="false" label="Show ASL videos" code="TDS_AS1" sortOrder="1"/>
  </Options>
</Tool>
```

Figure 26: Resulting XML for a Tool definition with 2 Tool Options.

Tool Dependencies

Tool Options themselves may optionally have a number of Tool Option Dependencies which can be expanded in a similar fashion to the Tool Options. Tool Option Dependencies are defined by the following trio of Input Variables:

- ToolOptionDependencyIfType_X_Y_Z
- ToolOptionDependencyIfCode_X_Y_Z
- ToolOptionDependencyDefault_X_Y_Z

Where again “X” is the Tool number and “Y” is the Option number, with “Z” defining the number of the Dependency - the values used for X, Y and Z must be sequential (e.g. 1, 2, 3, etc.). For the second Tool in a Tools group that has 2 Tool Option definitions, the first with 3 Dependencies and the second with 2 Dependencies, the Input Variables and values are shown in Figure 27.

ToolName_2	Masking
ToolType_2	
ToolStudentPackageFieldName_2	
ToolSortOrder_2	
ToolRequired_2	TRUE
ToolDisableOnGuest_2	FALSE
ToolAllowMultiple_2	FALSE
ToolAllowChange_2	TRUE
ToolOptionCode_2_1	TDS_Masking0
ToolOptionCode_2_2	TDS_Masking1
ToolOptionSortOrder_2_1	0
ToolOptionSortOrder_2_2	1
ToolOptionLabel_2_1	Masking Not Available
ToolOptionLabel_2_2	Masking Available
ToolOptionDefault_2_1	TRUE
ToolOptionDefault_2_2	FALSE
ToolOptionDependencyIfType_2_1_1	Language
ToolOptionDependencyIfCode_2_1_1	ENU
ToolOptionDependencyDefault_2_1_1	FALSE
ToolOptionDependencyIfType_2_1_2	Language
ToolOptionDependencyIfCode_2_1_2	ENU-Braille
ToolOptionDependencyDefault_2_1_2	FALSE
ToolOptionDependencyIfType_2_1_3	Language
ToolOptionDependencyIfCode_2_1_3	ESN
ToolOptionDependencyDefault_2_1_3	FALSE
ToolOptionDependencyIfType_2_2_1	Language
ToolOptionDependencyIfCode_2_2_1	ENU
ToolOptionDependencyDefault_2_2_1	TRUE
ToolOptionDependencyIfType_2_2_2	Language
ToolOptionDependencyIfCode_2_2_2	ESN
ToolOptionDependencyDefault_2_2_2	TRUE

Figure 27: Tool Option Dependency entries for the second Tool in a Tools group.

The resulting XML showing these values is shown in Figure 28.

```
<Tool name="Masking" allowChange="true" allowMultiple="false" sortOrder="2" disableOnGuest="false" required="true">
  <Options>
    <Option default="true" label="Masking Not Available" code="TDS_Masking0" sortOrder="0">
      <Dependencies>
        <Dependency default="false" ifToolType="Language" ifToolCode="ENU"/>
        <Dependency default="false" ifToolType="Language" ifToolCode="ENU-Braille"/>
        <Dependency default="false" ifToolType="Language" ifToolCode="ESN"/>
      </Dependencies>
    </Option>
    <Option default="false" label="Masking Available" code="TDS_Masking1" sortOrder="1">
      <Dependencies>
        <Dependency default="true" ifToolType="Language" ifToolCode="ENU"/>
        <Dependency default="true" ifToolType="Language" ifToolCode="ESN"/>
      </Dependencies>
    </Option>
  </Options>
</Tool>
```

Figure 28: The resulting XML from a Tool definition with multiple Options and multiple Dependencies.