Requirements Analysis

Smarter Balanced Assessment Consortium
Test Delivery System

Component: Test Scoring

Revision History

Revision Description		Date
Initial Release 1.0	David Lopez de Quintana	02/29/2016

Contents

Figures

Tables

1 Component Description

1.1 Description

The Test Scoring component is a new component that is not described by Smarter Balanced RFP-11 or in the Architecture document. The following diagram from the Architecture Report describes the expected logical interfaces.

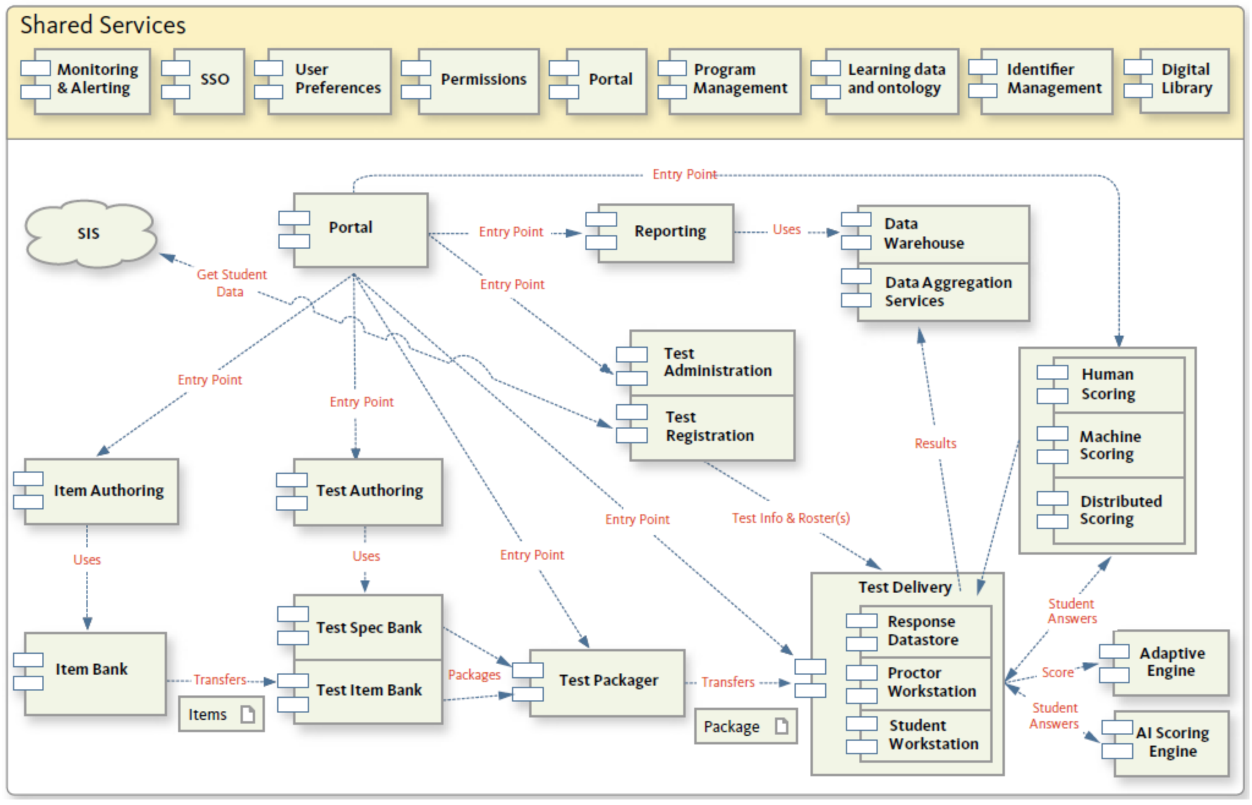


Figure 1. Figure 4.3 Logical Interfaces from the Architecture Report

Please note that assessment results are shown flowing directly from Test Delivery to the Data Warehouse. This flow is considered problematic because the downstream requirements of interfacing to external scoring engines such as human scoring and calculating overall test scale scores is incompatible with Test Delivery's mission of providing real time, interactive assessments to students. Once the interactive student portion of the assessment is over, Test Delivery needs to hand off the completed assessment to intermediate components to interface with external scoring engines to deliver responses and collect scores for items not scored during the assessment, and to fully scale score the completed assessment.

The new Test Integration component is being introduced to receive the completed assessment from Test Delivery, deliver responses to external scoring engines, receive and collate results from external scoring engines.

The new Test Scoring component is responsible for calculating the full scale scores for completed assessments. Test Scoring is required to be a separate component for Test Integration in the event that scale scores need to be displayed by the Test Delivery component at the end of the student assessment session and prior to receiving and collating results from all items not scored during the assessment.

1.2 Assessment Life Cycle

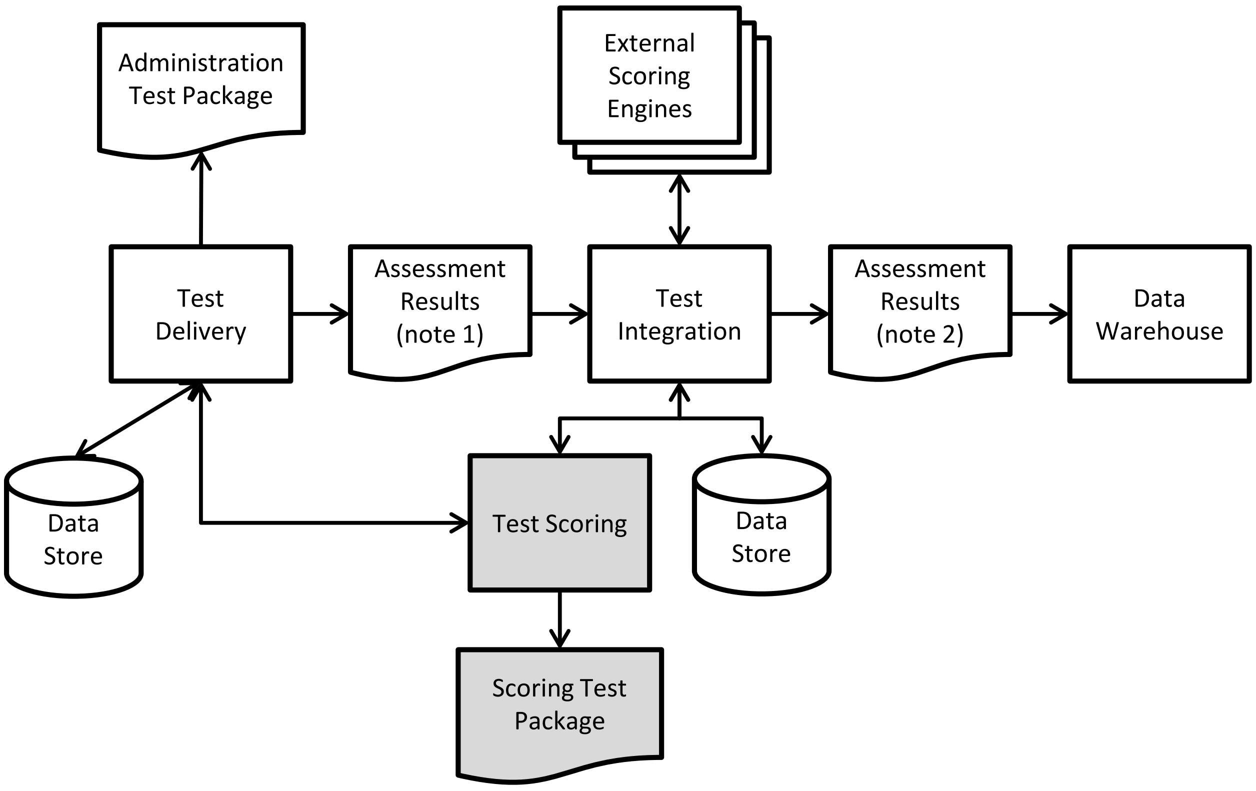
The phases of the assessment life cycle are detailed in pages 8 and 9 of the Architecture Report. The following is the description of the Test Administration and Scoring phases of the assessment life cycle:

> *Test Administration phase: This phase of the assessment life cycle includes the actual delivery of the assessment and the subprocesses contained within the phase. This includes proctoring, delivering, and collecting student responses.*
>
> *Scoring Phase: The scoring phase incorporates not only the actually scoring of student responses as well as any data needed for item statistics and trending. The Test Administration and the Test Registration components participate in the Pre-Test Administration phase of the assessment life cycle.*

This description includes calculation of scale scores during the Scoring phase, but does not include interfacing with external scoring engines and collating the results for items not scored during the assessment. This activity is also assumed to occur in the Scoring Phase.

1.3 Description

The following diagram displays the flow of assessment results from Test Delivery to the Data Warehouse. The shaded components are specific to Test Scoring.



NOTE 1: Contains responses and items scored during assessment, but missing items to be scored by external scoring engines and scale scores
NOTE 2: Complete with all item scores and scale scores

Figure 2. Assessment Results Flow

This assessment results flow modifies the flow in the Architecture document by inserting a new Test Integration component between Test Delivery and the Data Warehouse. Test Integration is responsible for the following functions:

1. Receiving and offloading completed assessments from Test Delivery. This allows Test Delivery to focus on its core mission of delivering real time, interactive assessments to students, and delivers the assessment results to Test Integration when that time-sensitive activity is complete.
2. Interfacing with external scoring engines that must score items, responses and rubrics that are designated for external scoring and not scored during the assessment by Test Delivery.
3. Holding an assessment while waiting for items scores from external scoring engines.
4. Integrating assessments with scores returned asynchronously from external scoring engines.
5. When all assessment items are scored and integrated, using the Test Scoring component to calculate assessment scale scores.
6. Delivering the completed assessment XML with all student information, accommodations, assessment event information, item responses, item scores and scale scores to the Data Warehouse.

The Test Scoring component is responsible for calculating scale scores based on the scored items in an assessment. Test Scoring is capable of scoring a partially scored assessment that is missing some item scores or student responses. This is done for the following reasons:

1. Some assessments may not require responses to all items. Therefore, some items may go unanswered by the student.
2. Test Scoring must be able to calculate preliminary scale scores on partially completed assessments where some items have no student responses, or partially scored assessments where items are awaiting scoring by external scoring engines. The main use case for this scenario is to retain the capability of presenting the student with a scale score at the end of the interactive assessment period for items

Scoring rules for a particular assessment requires that a Test Scoring component instance provide for the browsing of a Test Spec Bank for Scoring test packages, the selection of a particular test package, and the loading of the scoring rules into the Test Scoring component. This package should match the Administration test package that is similarly loaded into Test Delivery.

2 Functional Requirements from RFP-11

Please note that as new components, Test Integration and Test Scoring do not have any specific requirements allocated to it by RFP-11. However, the following RFP-11 requirements describe functionality that is selected for allocation to Test Scoring.

Req#	Requirement	Component Allocation	Discussion
	Adaptive Test Engine		
12.	System must be able to calculate preliminary scores using the raw scores produced from the machine scoring component and associated item score statistics from the Item Authoring and Pool application.	Test Scoring	*Not sure whether this requirement requires calculation of scale scores or just a raw test score. If it is for a raw test score, does it apply to Test Scoring?*

13.	System must be able to calculate final scores using ability estimate from machine scored items and results from human scored items and tasks.	Test Integration Test Scoring	The Test Integration component is responsible for integrating the machine scored items with results from human scored items and items scored by external scoring engines. The Test Scoring component is allocated the responsibility of calculating the scale scores for complete or partially complete assessments.
		Machine Scoring	
20.	System must be able to insert unscored items into tests and log them accordingly for research purposes.	Test Authoring Test Integration Test Scoring	This requirement indicates that some designated items may be left unscored by real-time scoring engines during test presentation and by any external scoring engines. Test Scoring will be able to calculate scale scores for partially completed assessments.
		Reporting	
92.	System will provide the ability (on/off system proctor preference) to display immediate preliminary score feedback to students at the conclusion of the test.	Test Scoring	This requirement indicates that the Test Delivery engine be able to interface directly with Test Scoring to calculate a preliminary scale score for an assessment at the conclusion of the student testing session.

3 Functional Requirements

	Requirement Category	**Functional Requirement**	**Description**
1.	Architecture	Test Scoring shall be an independent component that can be called from any component to calculate scale scores for an assessment.	Test Scoring is designed to be used to the Test Integration component to scale score assessments, but the Test Delivery may call it as well to scale score an assessment at the end of student testing. The interface to Test Scoring is independent of a specific calling component.
2.	Architecture	The Test Scoring component shall be written in Java and make use of open source technologies as required.	All Smarter Balanced Contract 11 components must be written in Java and utilize open source technologies.
3.	Architecture	The Test Scoring component shall use the Program Management shared service component to store configuration parameters.	Test Scoring relies on Smarter Balanced Contract 11 shared services.
4.	Architecture	The Test Scoring component shall use the Monitoring and Alerting shared service component to perform centralized logging.	Test Scoring relies on Smarter Balanced Contract 11 shared services.
5.	Configuration	Test Scoring shall provide a user interface for browsing test spec bank(s) for scoring test packages, selecting one or more scoring test packages, downloading the selected packages, and configuring itself using the content of the scoring test packages.	The endpoint URL for test spec banks to be browsed shall be stored in the Program Management component.
6.	Configuration	The format and content of the scoring test package shall conform to the rules specified in the *testpackage.dtd* document.	Completed and approved test packages are created by the Test Packaging function of Test Scoring and stored in a Test Spec Bank. The XML format of the packages stored in Test Spec Bank are documented by this XML DTD.

	Requirement Category	Functional Requirement	Description
7.	Scoring	Test Scoring shall be able to score assessment with scored as well as unscored items present.	The Test Scoring component is tolerant of partially completed assessments. It must compute scale scores for assessments where some item scores are not available. Items may be scheduled for scoring by external scoring engines, or may be designated to remain unscored according to RFP-11 requirement number 20.
8.	Scoring	Test Scoring shall be provided by the calling component with an assessment XML document containing student information, items and responses, and item scores.	The XML format for an assessment opportunity used in the interface for Test Scoring will be a known format and documented by a DTD. The provided XML will not contain scale scores, only item responses and scores.
9.	Scoring	Test Scoring shall return to the calling component a fully scale scored assessment by filling in scale score fields in the required XMLs with scale scores calculated according to the scoring rules provided by the configuration in the scoring test package.	Test Scoring fills in scale scoring fields in the DTD and returns to the calling component an XML document complete with scale scores for an assessment opportunity.
10.	Scoring	Test Scoring shall provide test scoring functions as described in section 4 Test Scoring Functions	The Test Scoring component is designed to implement specific test scoring functions as defined in this document. The Test Authoring component is configured to offer these scoring functions to test authors so they can instantiate them and provide the necessary parameters. These are provided to the Test Scoring component in the Scoring Test Package configuration XMLs described above.

4 Test Scoring Functions

For each test, scale and measure type a scoring rule can be defined. The order in which these scoring rules are run can be configured. The results from previously run rules (for the same test) can be used in subsequent scoring rules. Each scoring rule produces a score and a standard error. The score is either a number or a string (to allow things like "beyond proficient" or "Y"/"N" for attemptedness). Standard errors are always numeric but are optional.

Each scoring rule takes 2 standard parameters. These are:

1) `string measureOf`. The scale for which we are computing this measure. E.g. "Overall" if it is for the whole test, otherwise usually a strand name.
2) `string measureLabel`. Name of the measure to be computed. E.g. ScaleScore,PerformanceLevel.

Below follows the supported measure types and a list of available computation rules for each measure type.

4.1 Attemptedness Scoring Functions

4.1.1 SBACAttemptedness

Determine if the minimum number of items were attempted. Parameters:

Parameter	Type	Default	Description

Parameter	Type	Default	Description
MeasureOf	String	Overall	
MeasureLabel	String	Attempted	
TestPart	Dictionary<String, Integer>		Lookup with keys the segments on the test. The values are either 1 or 2. 1 means this segment is part of the CAT and 2 means it is part of the PERF.

Table 1. SBACAttemptedness Parameters

4.1.2 SBACIABAttemptedness

Parameter	Type	Default	Description
MeasureOf	String	Overall	
MeasureLabel	String	Attempted	

Table 2. SBACIBAAttemptedness Parameters

4.1.3 SBACBlockAttemptedness

Parameter	Type	Default	Description
MeasureOf	String		Block name, e.g. SOCK_LT_68.
MeasureLabel	String	Attempted	
TestPart	Dictionary<String, Integer>		Lookup with keys the segments making up this block (the segments that correspond to the component test for this block). Values are always 1.

Table 3. SBACBlockAttemptedness Parameters

4.2 Item Counts

4.2.1 ItemCount

Parameter	Type	Default	Description
Subscale	String		”Overall” or a reporting category
MeasureLabel	String	ItemCount	

Table 4. ItemCount Parameters

4.2.2 MultipleStrandItemCount

Parameter	Type	Default	Description
Subscale	String		Reporting category
MeasureLabel	String	ItemCount	
Subscales	Dictionary<Integer, String>		Array of strands that make up the reporting category (keys are 1,2, etc. Values are the strands)

Table 5. MultipleStrandItemCount Parameters

4.2.3 ItemCountScored

Parameter	Type	Default	Description

Parameter	Type	Default	Description
Subscale	String		”Overall” or a reporting category
MeasureLabel	String	ItemCountScored	

Table 6. ItemCountScored Parameters

4.2.4 MutipleStrandItemCountScored

Parameter	Type	Default	Description
Subscale	String		Reporting category
MeasureLabel	String	ItemCountScored	
Subscales	Dictionary<Integer, String>		Array of strands that make up the reporting category (keys are 1,2, etc. Values are the strands)

Table 7. MultipleStrandItemCountScored Parameters

4.3 Raw Scores

4.3.1 RawScore

Parameter	Type	Default	Description
Subscale	String		”Overall” or a reporting category
MeasureLabel	String	RawScore	

Table 8. RawScore Parameters

4.3.2 MultipleStrandRawScore

Parameter	Type	Default	Description
Subscale	String		Reporting category
MeasureLabel	String	RawScore	
Subscales	Dictionary<Integer, String>		Array of strands that make up the reporting category (keys are 1,2, etc. Values are the strands)

Table 9. MultipleStrandItemRawScore Parameters

4.4 Theta Scoring Functions

4.4.1 SBACTheta

Parameter	Type	Default	Description
MeasureOf	String		”Overall” or a reporting category
measureLabel	String	ThetaScore	
LOT	Float		Lowest obtainable theta
HOT	Float		Highest obtainable theta

Parameter	Type	Default	Description
seLimit	Float	2.5	Max standard error (on theta scale)

Table 10. SBACTheta Parameters

4.4.2 SBACCATTheta

Parameter	Type	Default	Description
MeasureOf	String		reporting category
measureLabel	String	ThetaSc ore	
LOT	Float		Lowest obtainable theta
HOT	Float		Highest obtainable theta
seLimit	Float	2.5	Max standard error (on theta scale)
averageA	Float		Average a parameter for items in this subject and grade.
averageB	Float		Average b parameter for items in this subject and grade.

Table 11. SBACCATTheta Parameters

4.4.3 SBACMultiStrandTheta

Parameter	Type	Default	Description
MeasureOf	String		reporting category
measureLabel	String	ThetaSc ore	
LOT	Float		Lowest obtainable theta
HOT	Float		Highest obtainable theta
seLimit	Float	2.5	Max standard error (on theta scale)
strands	Dictionary<Int eger, String>		Array of strands that make up this reporting category.

Table 12. SBACMultiStrandTheta Parameters

4.4.4 SBACCATMultiStrandTheta

Parameter	Type	Default	Description
MeasureOf	String		reporting category
measureLabel	String	ThetaSc ore	
LOT	Float		Lowest obtainable theta
HOT	Float		Highest obtainable theta
seLimit	Float	2.5	Max standard error (on theta scale)
strands	Dictionary<Int eger, String>		Array of strands that make up this reporting category.
averageA	Float		Average a parameter for items in this subject and grade.
averageB	Float		Average b parameter for items in this subject and grade.

Table 13. SBACCATMultiStrandTheta Parameters

4.4.5 SBACMultiSegmentTheta

Parameter	Type	Default	Description
MeasureOf	String		A block name on an IAB test

Parameter	Type	Default	Description
measureLabel	String	ThetaScore	
LOT	Float		Lowest obtainable theta
HOT	Float		Highest obtainable theta
seLimit	Float	2.5	Max standard error (on theta scale)
segments	Dictionary<String, Integer>		Lookup with keys the segments making up this block (the segments that correspond to the component test for this block). Values are always 1.

Table 14. SBACMultiSegmentTheta Parameters

4.5 Scale Score

4.5.1 ScaleScore

Parameter	Type	Default	Description
MeasureOf	String		”Overall” or a reporting category
MeasureLabel	String	ScaleScore	

Table 15. ScaleScore Parameters

4.6 Block Score Functions

4.6.1 SBACNumBlocks

Parameter	Type	Default	Description
MeasureOf	String	Overall	
MeasureLabel	String	NumberBlocks	
Blocks	Dictionary<Integer, String>		Array of block names on this IAB test.

Table 16. SBACNumBlocks Parameters

4.6.2 SBACNumBlocksProficient

Parameter	Type	Default	Description
MeasureOf	String	Overall	
MeasureLabel	String	NumBlocksAboveStandard	
Blocks	Dictionary<Integer, String>		Array of block names on this IAB test.

Table 17. SBACNumBlocksProficient Parameters

4.7 Performance Level

4.7.1 SEBasedPerformanceIndicator

Parameter	Type	Default	Description
MeasureOf	String		Reporting category
MeasureLabel	String	PerformanceLevel	
seMultiple	Float	1.5	Multiple of SE used to construct the middle performance level.
ProficientPerformanceLevel	Integer	3	Overall performance cut that corresponds to the proficient cut.

Table 18. SEBasedPerformanceIndicator Parameters

4.7.2 SEBasedPLWithoutRounding

Parameter	Type	Default	Description
MeasureOf	String		Reporting category
MeasureLabel	String	PerformanceLevel	
seMultiple	Float	1.5	Multiple of SE used to construct the middle performance level.
ProficientPerformanceLevel	Integer	3	Overall performance cut that corresponds to the proficient cut.
LOT	Float		Lowest obtainable theta
HOT	Float		Highest obtainable theta

Table 19. SEBasedPLWithoutRounding Parameters

4.7.3 TestPerformanceLevel

Parameter	Type	Default	Description
MeasureOf	String	Overall	
MeasureLabel	String	PerformanceLevel	

Table 20. TestPerformanceLevel Parameters

4.8 AccommodationUseCodes

4.8.1 SBACAccommodationUseCodes

Parameter	Type	Default	Description
MeasureLabel	String	Overall	
MeasureOf	String	Accommodation Codes	
AccomNoCodes	Dictionary<String, String>		Key is Accommodation Type and the value is the code that corresponds to "No"

Table 21. SBACAccommodationUseCodes Parameters