Scenario #: 3

Scenario Description: Storage of Items and Related Resources

- Store items and related resources
- Parse and store items which share resource files
- Maintain item version information

Version Control

Version #	Date	Author	Description
1.0	01/15/2013	Russ Hammond	Initial Draft
1.1	01/25/2013	Russ Hammond	Altered to check test results using Test Item Bank rather than using database queries.

Test Scripts

The following scripts will cover this scenario:

- 3.1 Store item which has an associated resource file
- 3.2 Parse and store items which share a common resource file
- 3.3 Parse and store items which share a common resource file (part 2)
- 3.4 Reject an item with a version less than or equal to that currently stored.
- 3.5 Runtime exception should be sent to Monitoring and Alerting

Test Components/Requirements

This test scenario covers the following high-level test requirements (see scripts below for specific requirements covered by each test script):

- Test Item Bank
- Test Item Bank Requirements RADTIB.1.x, RADTIB.3.x, PRTIB.1.1

User Groups

- Item Bank Authors

Script #: 3.1 – Store an Item with an Associated Resource

Script Description

- Allow imported items to be associated to resources by XML configuration in the imsmanifest.xml file.
- **This script should be executed concurrently with Script 2.7.**

Testing Requirements

This test script covers the following specific testing requirements:

- RADTIB.1.1
- RADTIB.1.1.2
- RADTIB.1.1.4
- RADTIB.1.3
- RADTIB.1.4

Setup

- Test Item Bank must be deployed on CloudFoundry, with MongoDB operational
- QA staff must have Rest Client available to perform test
- Create a package of items in which some items have associated resource files. The resource files should not be shared by multiple items.

Teardown

- None – imported items can be retained if the test is successful.

Step #	Test Action	Expected Results	Pass/ Fail
1	Access the Rest Client	Rest Client viewed in Chrome Browser	
2	Using the REST Client, access the API to import items http://{server-context}/sftpFileImport Refer to Script 2.7.	The API for the component is displayed.	
3	Import the test package of items	• Items SQA080-085, 87 and 89 are stored with resource files. • Items are stored	

Step #	Test Action	Expected Results	Pass/ Fail
		• Searchable metadata is stored • Resource files are stored • Resource files are associated with correct items • An alert should be issued which says 8 items were added	
4	Save Import Set number from step 3. Execute API http://{server-context}/importSet/{ImportSet#}	• Time and status of the import will be displayed	
5	Search for the item in the database using http://{server-context} /item/apipitem_SQA001	• Metadata for item will be displayed.	
6	Export the items SQA080-085, 87 and 89 using http://{server-context} /exportSet Sample JSON input `{ "items": [ { "identifier": "I_00001_QR", "version": "1.1" }, { "identifier": "I_00002_QR", "version": "1.2" } ] }`	• Items will be exported into zip files. Zip files will contain the item, the associated resource, and the metadata file.	

Test Execution

Date/Time	Tester	Test ID	Test Phase	Status
1/30/2013	RDH		1	Passed
1/30/2013	RM		1	Passed

Script #: 3.2 – Parse and store items which share a common resource file

Script Description

- Allow multiple imported items to be associated to a common resource by XML configuration in the imsmanifest.xml file.

Testing Requirements

This test script covers the following specific testing requirements:

- RADTIB.1.1
- RADTIB.1.1.2
- RADTIB.1.1.4
- RADTIB.1.3
- RADTIB.1.4
- RADTIB.1.10

Setup

- Test Item Bank must be deployed on CloudFoundry, with MongoDB operational
- QA staff must have Rest Client available to perform test
- Create a package of items in which some items share a resource file.

Teardown

- None – imported items can be retained if the test is successful.

Step #	Test Action	Expected Results	Pass/ Fail
1	Access the Rest Client	Rest Client viewed in Chrome Browser	
2	Using the REST Client, access the API to import items http://{server-context}/sftpFileImport Use folder Script 3.2	The API for the component is displayed.	
3	Import the test package of items	• Items are stored • Resource file **me.png** is stored • Searchable metadata is stored	

Step #	Test Action	Expected Results	Pass/ Fail
		• All 10 added items are associated to resource **me.png** • An alert is issued which says 10 items were added	
4	Save Import Set number from step 3. Execute API http://{server-context}/importSet/{ImportSet#}	• Time and status of the import will be displayed	
5	Search for the item in the database using http://{server-context} /item/apipitem_SQA001	• Metadata for item will be displayed.	
6	Export the items SQA300-309 http://{server-context} /exportSet Sample JSON input `{ "items": [ { "identifier": "I_00001_QR", "version": "1.1" }, { "identifier": "I_00002_QR", "version": "1.2" } ] }`	• Items will be exported into zip files. Zip files will contain the item, the associated resource, and the metadata file.	
		•	

Test Execution

Date/Time	Tester	Test ID	Test Phase	Status
1/30/2013	RDH		1	Passed

1/30/2013	RM		1	Passed

Script #: 3.3 – Parse and store valid items which share a common resource file (part 2)

Script Description

- Allow multiple imported items to be associated to a common resource by XML configuration in the imsmanifest.xml file.. Import only those items that do not fail other validation criteria.

Testing Requirements

This test script covers the following specific testing requirements:

- RADTIB.1.1
- RADTIB.1.1.2
- RADTIB.1.1.4
- RADTIB.1.3
- RADTIB.1.4
- RADTIB.1.10

Setup

- Test Item Bank must be deployed on CloudFoundry, with MongoDB operational
- QA staff must have Rest Client available to perform test
- Create a package of items in which some items share a resource file. Some of the items with the shared resource should be valid, and some should fail other criteria, such as missing metadata or missing a Metadata File.

Teardown

- None – imported items can be retained if the test is successful.

Step #	Test Action	Expected Results	Pass/ Fail
1	Access the Rest Client	Rest Client viewed in Chrome Browser	
2	Using the REST Client, access the API to import items http://{server-context}/sftpFileImport Use folder Script 3.3	The API for the component is displayed.	

Step #	Test Action	Expected Results	Pass/ Fail
3	Import the test package of items	• Valid items are stored • Resource file **me.png** is stored • Resource files are associated with correct items • An alert is generated to say 8 items have been added • Invalid items should be rejected • SQA311 is missing metadata file • SQA316 is missing item type metadata • An alert should be generated for each rejected item.	
4	Save Import Set number from step 3. Execute API http://{server-context}/importSet/{ImportSet#}	• Time and status of the import will be displayed	
5	Search for the item in the database using http://{server-context} /item/apipitem_SQA###	• Metadata for stored items will be displayed. SQA310, 312-315, 317-319 should be available.	
6	Export the items successfully added. http://{server-context} /exportSet Sample JSON input `{ "items": [ { "identifier": "I_00001_QR", "version": "1.1" }, { "identifier": "I_00002_QR", "version": "1.2" } ]`	• Items will be exported into zip files. Zip files will contain the item, the associated resource, and the metadata file.	

Step #	Test Action	Expected Results	Pass/ Fail
	}		

Test Execution

Date/Time	Tester	Test ID	Test Phase	Status
1/30/2013	RDH		1	Passed
1/30/2013	RM		1	Passed

Script #: 3.4 – Reject an Item which has a version number less than or equal to the currently stored item.

Script Description

- Allow multiple imported items to be associated to a common resource by XML configuration in the imsmanifest.xml file.

Testing Requirements

This test script covers the following specific testing requirements:

- RADTIB.1.1
- RADTIB.1.1.2
- RADTIB.1.1.4
- RADTIB.1.3
- RADTIB.1.4
- RADTIB.1.10
- RADTIB.3.3

Setup

- Test Item Bank must be deployed on CloudFoundry, with MongoDB operational
- QA staff must have Rest Client available to perform test
- Create a package of items in which some items have a version number not greater than the currently stored item.

Teardown

- None – imported items can be retained if the test is successful.

Step #	Test Action	Expected Results	Pass/ Fail
1	Access the Rest Client	Rest Client viewed in Chrome Browser	
2	Using the REST Client, access the API to import items http://{server-context}/sftpFileImport Use folder Script 3.4	The API for the component is displayed.	
3	Import the test package of items	• Items with greater version numbers are successfully	

Step #	Test Action	Expected Results	Pass/ Fail
		imported and stored. • Items with versions less than or equal to the current item version are rejected and an alert is created. • Items SQA301, 303, 304 and 309 fail due to version number • An alert is created to say how 6 were added to the Test Item Bank	
4	Save Import Set number from step 3. Execute API http://{server-context}/importSet/{ImportSet#}	• Time and status of the import will be displayed	
5	Search for the item in the database using http://{server-context} /item/apipitem_SQA###	• Metadata for stored items will be displayed. New versions of SQA300, 302, 305-308 should be available. Old versions of 301, 303, 304 and 309 should still be available.	

Test Execution

Date/Time	Tester	Test ID	Test Phase	Status
1/30/2013	RDH		1	Passed
1/30/2013	RM		1	Passed

Script #: 3.5 – Runtime exception should be routed to Monitoring and Alerting

Script Description

- Allow multiple imported items to be associated to a common resource by XML configuration in the imsmanifest.xml file.

Testing Requirements

This test script covers the following specific testing requirements:

- RADTIB.1.1
- RADTIB.1.1.2
- RADTIB.1.1.4
- RADTIB.1.3
- RADTIB.1.4
- RADTIB.1.10
- RADTIB.3.3
- PRTIB.1.1

Setup

- Test Item Bank must be deployed on CloudFoundry, with MongoDB operational
- QA staff must have Rest Client available to perform test
- Create a package of items in which some items have a version number not greater than the currently stored item.
- Script 3.5, SQA301 has non-numeric version

Teardown

- None – imported items can be retained if the test is successful.

Step #	Test Action	Expected Results	Pass/ Fail
1	Access the Rest Client	Rest Client viewed in Chrome Browser	
2	Using the REST Client, access the API to import items http://{server-context}/sftpFileImport	The API for the component is displayed.	

Step #	Test Action	Expected Results	Pass/ Fail
	Use folder Script 3.5		
3	Import the test package of items	• Items with greater version numbers are successfully imported and stored. • SQA302 is stored • Items with versions less than or equal to the current item version are rejected and an alert is created. • SQA300 is rejected due to version • An alert is created to say 1 items were added to the Test Item Bank • SQA301 has a non-numeric version, expect it to throw a runtime exception • An alert should be created for the runtime exception	
4	Save Import Set number from step 3. Execute API http://{server-context}/importSet/{ImportSet#}	• Time and status of the import will be displayed	
5	Search for the item in the database using http://{server-context} /item/apipitem_SQA###	• Metadata for stored items will be displayed. New versions of SQA302 should be available. Old versions of 300, 301should still be available.	

Test Execution

Date/Time	Tester	Test ID	Test Phase	Status
1/30/2013	RDH		1	Passed
1/30/2013	RM		1	Passed