ECS Performance Test Cases

Table of Contents

1 BMGT 1
1.1 Performance 1
1.1.1 Performance - Aggregate (ECS-ECSTC-564) 1
1.1.2 Performance - Granule Inserts/Deletes (ECS-ECSTC-565) 7
1.1.3 Performance - QA Updates (ECS-ECSTC-566) 10
1.1.4 Performance - Datapool Inserts/Updates/Deletes (ECS-ECSTC-567) 14
1.1.5 Performance - Granule Updates (ECS-ECSTC-568) 17
1.1.6 Performance - Collection Inserts/Updates (ECS-ECSTC-569) 22
1.1.7 Performance - MISBR (ECS-ECSTC-570) 24
1.1.8 Performance - Long Form Verification (ECS-ECSTC-571) 29

List of Figures

No table of figures entries found.

List of Tables

No table of figures entries found.

1 BMGT

1.1 Performance

1.1.1 Performance - Aggregate (ECS-ECSTC-564)

DESCRIPTION:

	S	670	1	**[Performance – Aggregate]** Perform the following operations, ensuring that different granules are used for each: a) 10,000 granule inserts b) 25,000 granule deletions c) 50,000 QA updates d) 10,000 DPL publications e) 25,000 DPL unpublications f) 5,000 other granule updates (e.g. hide/unhide, restriction, etc) g) 50 collection inserts or updates h) 5,000 browse inserts and linkages		

PRECONDITIONS:

STEPS:

#	Action	Expected Result	Notes
1	Pre-Conditions		#comment
2	Configure BMGT to export either to a fake ECHO endpoint, or to a proxy which will allow logging of all exports.		
3	Ensure the collections are installed and enabled for collection and granule export. This can be confirmed on the BMGT GUI (Collection Configuration tab) or via the database as follows: SELECT granuleexportflag , collectionexportflag FROM bg_collection_configuration WHERE shortname = <SHORTNAME> AND versionid = <VERSIONID>; If they are not enabled, UPDATE bg_collection_configuration SET granuleexportflag = 'Y', collectionexportflag = 'Y' WHERE shortname = <SHORTNAME> AND versionid = <VERSIONID>;<br		

#	Action	Expected Result	Notes
	/> Allow any export requests generated by these actions to complete.		
4	Ensure that the BMGT system is quiesced, with no requests in the PENDING state.		
5	Pause the dispatcher using the GUI		
6	Ensure that the automatic driver is running		
7	Setup		#comment
8	S-1 Perform the following operations, ensuring that different granules are used for each:		#comment
9	a) 10,000 granule inserts		#comment
10	Find 10,000 granules.		
11	Write the geoids of all 10,000 granules to a file. Use a psql command and direct the output into a file: >psql -h p4dbl03 -U readonly -d ecs -c "SELECT g.granuleid, g.shortname, g.versionid FROM aim.amgranule g JOIN aim.ambrowsegranulexref x ON g.granuleid=x.granuleid JOIN aim.bg_collection_configuration b ON g.collectionid = b.collectionid WHERE g.shortname ='${C1.ShortName}' AND g.VersionId = ${C1.VersionId} AND g.deleteeffectivedate IS NULL AND b.granuleexportflag = 'Y' AND b.collectionexportflag = 'Y' LIMIT 10000" -q | egrep "[0-9]+" | grep -v rows | awk '{printf "SC:%s.%03d:%d\n", $3, $5, $1;}' > geoids. (substituing the correct values for ${C1.ShortName} and ${C1.VersionId} 		
12	Using MOD10A1.086 for the manual export		#comment
13	Perform a manual export of all 25,000 granule inserts: EcBmBMGTManualStart <MODE> --metg --granulefile <granuleIdFile> 		
14	b) 25,000 granule deletions		#comment

15	Find 25,000 non existent granule IDs. To do this:
	select max(granuleid) from amgranule
	then multiply this number by 10. This will be the base id to use.
	
	run this perl script:
	
	$base = 100;
	$id = $base;
	$gransleft = 25000;
	while ($gransleft-- > 0){
	print "SC:<SNVI>:" . $id++ ."\n";
	}
	
	Where $base is set to the base ID determined above, and the correct SNVI for collection C2 is subsititued in place of "<SNVI>"
	Redirect the output of this script to a file.		

16	Using MOD11_L2.004 for deletes.		#comment
17	Perform a manual export of all 25,000 granule deletes:<br		

#	Action	Expected Result	Notes
	/>EcBmBMGTManualStart <MODE> -del --metg --granulefile <granuleIdFile> 		
18	c) 50,000 QA updates		#comment
19	Find 50,000 distinct granules in the same collection: select granuleId from AmGranule where ShortName = '${C3.ShortName}' and VersionId = ${C3.VersionId} limit 50000		
20	Add the granuleids to a file QA_GRANULEID_FILE		
21	Record the time as QA_START_TIME		
22	See the QAUU 609 Section 4.8.9: http://edhs1.gsfc.nasa.gov/waisdata/eed/pdf/609eed001_Rev01_File_7.pdf		#comment
23	Find one granule's metadata file: select p.path || '/' || f.archivemetfilename from ammetadatafile f join dsmdxmlpath p on f.archivepathid = p.archivepathid where f.granuleid = ${GRANULEID}		
24	Find the name of a measured parameter to update: xpath ${XML_FILE} '//MeasuredParameterContainer/ParameterName/text()'		
25	Find the measured parameter's science quality flag value: xpath ${XML_FILE} "//MeasuredParameterContainer[ParameterName='${PARAMETER_NAME}']/QAFlags/ScienceQualityFlag/text()"		
26	Select a different science quality flag value from the list of valid values in the properties file: grep VALID_SCIENCE_QA_FLAGS /usr/ecs/${MODE}/CUSTOM/cfg/EcDsAmQaUpdateUtility.properties		
27	Note that the above step needs to be done on p5dpl01 in PVC OPS mode.		#comment
28	Found out the that first granule (3085987951) had "Not Investigated" as flag for parameter "Ice Surface Temperature". Will choose "Hold" as new flag.		#comment
29	Create a QA update request file named according to the the QAUU 609: <MODE>_<Site>_QAUPDATE<description>.<YYYY><MM><DD><hh><mm><ss> E.g., OPS_BMGT82_QAUPDATE_BE_82_01_Crit_687.20130522231600		
30	Write a QA update request to update a science flag, using a GranuleUR (replace <TAB> with an actual tab character): First, add the		

#	Action	Expected Result	Notes
	file header: From ${site} begin QAMetadataUpdate Science GranuleUR		
31	Add the file contents: cat QA_GRANULEID_FILE | awk '{print "${shortname}<TAB>${versionid}<TAB>${shortname}.${versionid}:" $1 "<TAB>${measured_parameter_name}<TAB>${qa_flag_value}<TAB>${qa_flag_explanation}" >> REQUEST_FILE		
32	Actually this works rather than the previous step: > cat QA_GRANULEID_FILE | awk '{printf "%1$s\t86\tSC:%1$s.%2$03d:%3$d\tALL\t%4$s\t%5$s\n", "<shortname>", <version id>, $1, "<quality flag>", "BmgtPerfTest_<request file>";}' > OPS_LaRC_QAUU.20130522231600		#comment
33	Add the file footer: end QAMetadataUpdate		
34	Copy the QA update request file to the QA request directory configured in the properties file: sed -n 's/QA_REQUEST_DIR *= *//p' /usr/ecs/${MODE}/CUSTOM/cfg/EcDsAmQaUpdateUtility.properties		
35	Run QAUU, specifying the filename (but not the path) if there are other request files in the request directory: EcAmQAUUStart ${MODE} -file ${REQUEST_FILE} -noprompt		
36	Verify that the QA Updates resulted in events in the DsMdGrEventHistory table: Select count(1) from DsMdGREventHistory where eventtime > {QA_START_TIME} and eventType = 'GRQAUPDATE' This should show 50,000 events		
37	d) 10,000 DPL publications		#comment
38	select 10,000 non public granules from collection C4 and write their IDs to a file select granuleid from AmGranule where ShortName = ${C4.ShortName} and VersionId = ${C4.VersionId} and coalesce(isOrderOnly,'B') in ('Y', 'H') limit 10000 Write results to a file		
39	Actually the following query should be used to find 10K non-public granules to publish: select granuleid from AmGranule where ShortName = ${C1.ShortName} and VersionId = ${VersionId} and coalesce(isOrderOnly,'B') in ('Y', 'H') and coalesce(DeleteFromArchive, 'N') = 'N' and DeleteEffectiveDate IS NULL and granuleid not in (select granuleid from DsMdGranuleRestriction) limit 10000;		#comment

#	Action	Expected Result	Notes
40	publish the granules. EcDlPublishUtilityStart <MODE> -ecs -file <FileName>		
41			#comment
42	e) 25,000 DPL unpublications		#comment
43	select 25,000 public granules from collection C5 and write their IDs to a file select granuleid from AmGranule where ShortName = ${C5.ShortName} and VersionId = ${C5.VersionId} and coalesce(isOrderOnly,'B') ='B' limit 25000 Write results to a file		
44	unpublish the granules. EcDlUnpublishStart -mode <MODE> -file <FileName>		
45	f) 5,000 other granule updates (e.g. hide/unhide, restriction, etc)		#comment
46	Ensure that there are enough granules in the collection by ensuring that this query returns a number higher than 5000: select count(1) from AmGranule where granuleId not in (select granuleid from DsMdGranuleRestriction) and shortname = ${C6.ShortName} and versionid = ${C6.VersionId} 		
47	Find a restriction flag value to use. valid values are found by running: select * from DsMdRestrictionFlag where restrictionFlag != 255 note one of the RestrictionFlag column values and record it as ${RestrictionFlag} If no rows are returned, insert a new row with restrictionFlag value < 255 and > 0		
48	Insert the restrictions: insert into DsMdGranuleRestriction (granuleid, restrictionflag, unpublishflag, lastupdate) values select granuleid, ${RestrictionFlag}, 'N', lastupdate from AmGranule where granuleId not in (select granuleid from DsMdGranuleRestriction) and shortname = ${C6.ShortName} and versionid = ${C6.VersionId} limit 5000		
49	Actually we used this query: insert into dsmdgranulerestriction (granuleid, restrictionflag, unpublishflag, lastupdate) select granuleid, 254, 'N', lastupdate from amgranule where granuleid not in (select granuleid from DsMdGranuleRestriction) and shortname = 'MYD13Q1' and versionid = 86 limit 5000;		#comment
50	g) 50 collection inserts or updates		#comment
51	Update the DsGeESDTConfiguredType table manually to simulate collection updates by executing the following queries sequentially: update DsGeESDTConfiguredType set esdtstate = 'updating' limit 50;<br		

#	Action	Expected Result	Notes
	/> update DsGeESDTConfiguredType set esdtstate = 'installed' where esdtstate = 'updating';		
52	h) 5,000 browse inserts and linkages		#comment
53	Identify 1 browse image to use for this test (since it is not normal to have so many granules linked to the same browse, it would be a good idea to unlink after completing this test)		
54	Add Browse linkages to the browse Xref table: insert into AmBrowseGranuleXref (granuleId, browseId) values select granuleId, ${BrowseId} from AmGranule where ShortName = '${C7.ShortName}' and VersionId = ${C7.VersionId}		
55	We used this query: insert into ambrowsegranulexref (granuleid, browseid) select granuleid, 2035880617 from amgranule where shortname = 'MISQA' and versionid = 3;		#comment
56	Start the export		#comment
57	Ensure that the BMGT Automatic driver is running or start it.		
58	Ensure that export requests have been generated in the bg_export_request table for all of the actions above		
59	Resume BMGT dispatcher using the GUI, recording the time at which the resumption occurs		
60	Verification		#comment
61	V-1 Verify that the operations in S-1 result in the export of the following:		#comment
62	a) 105,000 HTTP PUT requests containing full granule metadata		#comment
63	Inspect the proxy or fake ECHO log to verify that 105,000 Granule HTTP PUT requests were sent.		
64	b) 25,000 HTTP DELETE requests sent to URLs reflecting granule resources, but containing no request body.		#comment
65	Inspect the proxy or fake ECHO log to verify that 25,000 Granule HTTP DELETE requests were sent.		
66	c) 50 HTTP PUT requests containing full collection metadata.		#comment
67	Inspect the proxy or fake ECHO log to verify that 50 Collection HTTP PUT requests were sent.		
68	V-2 Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation. TBD		#comment
69	V-3 Verify that all exports resulting from S-1 complete within 1 hour of		#comment

#	Action	Expected Result	Notes
	the start of the test.</i>		
70	Verify that all 130,050 requests are moved to the SUCCESS state within 1 hour after the recorded dispatcher resumption time.		

TEST DATA:

Crit id	Crit ccr no	Test Data Description	Data Type Requirements	Metadata Requirements	Volume Requirements	Size Requirements	Data Location	Readiness Status

EXPECTED RESULTS:

	V	670	1	Verify that the operations in S-1 result in the export of the following: a) 105,000 HTTP PUT requests containing full granule metadata b) 25,000 HTTP DELETE requests sent to URLs reflecting granule resources, but containing no request body. c) 50 HTTP PUT requests containing full collection metadata.		
	V	670	2	Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		
	V	670	3	Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		

1.1.2 Performance - Granule Inserts/Deletes (ECS-ECSTC-565)

DESCRIPTION:

	S	680	1	**[Performance - Granule Inserts/Deletes]** Perform the following operations, ensuring that different granules are used for each: a) 25,000 granule inserts for granules with ECS metadata which are also linked to browse granules. b) 25,000 granule deletes		

PRECONDITIONS:

STEPS:

#	Action	Expected Result	Notes
1	Pre-Conditions		#comment
2	Configure BMGT to export either to a fake ECHO endpoint, or to a proxy which will allow logging of all exports.		
3	Ensure the collections are installed and enabled for collection and granule export. This can be confirmed on the BMGT GUI (Collection Configuration tab) or via the database as follows: SELECT granuleexportflag , collectionexportflag FROM bg_collection_configuration WHERE shortname = <SHORTNAME> AND versionid = <VERSIONID>; If they are not enabled, UPDATE bg_collection_configuration SET granuleexportflag = 'Y', collectionexportflag = 'Y' WHERE shortname = <SHORTNAME> AND versionid = <VERSIONID>; 		
4	Ensure that the BMGT system is quiesced, with no requests in the PENDING state.		
5	Pause the dispatcher using the GUI (System Status tab)		
6	Setup		#comment
7	S-1 Perform the following operations, ensuring that different granules are used for each: a) 25,000 granule inserts for granules with ECS metadata which are also linked to browse granules.		#comment
8	Find 25,000 granules which have browse linkages.		
9	Write the geoids of all 25,000 granules to a file.		
10	Use a psql command and direct the output into a file: >psql -h p4dbl03 -U readonly -d ecs -c "SELECT g.granuleid, g.shortname, g.versionid FROM aim.amgranule g JOIN aim.ambrowsegranulexref x ON g.granuleid=x.granuleid JOIN aim.bg_collection_configuration b ON g.collectionid = b.collectionid WHERE g.shortname NOT LIKE 'ECS%' AND g.deleteeffectivedate IS NULL AND b.granuleexportflag = 'Y' AND b.collectionexportflag = 'Y' LIMIT 25" -q	egrep "[0-9]+"	grep -v rows	awk '{printf "SC:%s.%03d:%d\n", $3, $5, $1;}' > geoids. Alternatively, a script to find suitable granules is in /tools/common/test/new_scripts/FindDataForBMGT_performance.sh. The result is placed in /usr/ecs/OPS/CUSTOM/utilities with a filename of the form BmgtGeoid_perf_<date/time>.txt.		#comment
11	Perform a manual export of all 25,000 granule inserts:<br		

#	Action	Expected Result	Notes
	/>EcBmBMGTManualStart <MODE> --metg --granulefile <granuleIdFile> 		
12	S-1 Perform the following operations, ensuring that different granules are used for each: b) 25,000 granule deletes		#comment
13	Generate 25,000 non existent granule IDs. From a database command prompt, use: SELECT MAX(granuleid) FROM amgranule; to find the largest granule ID, then multiply this number by 10. This will be the base id to use. From a command prompt: > seq 1 25000 | awk '{printf "SC:MOD11_L2.004:%d\n", $1 + <base id>;}' > delete_geoids 		
14	I used MOD11_L2.005. File is in /tools/common/test/new_scripts/BMGTperf/nonexist.out		#comment
15	Perform a manual export of all 25,000 granule deletes: EcBmBMGTManualStart <MODE> -del --metg --granulefile <granuleIdFile>		
16	Resume BMGT dispatcher using the GUI, recording the time at which the resumption occurs		
17	Verification		#comment
18	V-1 Verify that the operations in S-1 result in the export of the following: a) 25,000 HTTP PUT requests containing full granule metadata		#comment
19	Inspect the proxy or fake ECHO log to verify that 25,000 HTTP PUT requests were sent. > cd /L0_buffer/OPS/ProductOutput/FakedEcho82/logs > grep PUT access.log | grep '2013-07-02T15' | wc -l Note: You will need to modify the grep commands to match the data or timestamps used.		
20	V-1 Verify that the operations in S-1 result in the export of the following: b) 25,000 HTTP DELETE requests sent to URLs reflecting granule resources, but containing no request body.		#comment
21	Inspect the proxy or fake ECHO log to verify that 25,000 HTTP DELETE requests were sent. > cd /L0_buffer/OPS/ProductOutput/FakedEcho82/logs > grep DELETE access.log | grep MOD11_L2\.004 | wc -l Note: You will need to modify the grep commands to match the data or timestamps used.		
22	V-2 Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		#comment
23	TBD		

#	Action	Expected Result	Notes
24	V-3 Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		#comment
25	Verify that all 50,000 requests are moved to the SUCCESS state within 1 hour after the recorded dispatcher resumption time.		

TEST DATA:

Crit id	Crit ccr no	Test Data Description	Data Type Requirements	Metadata Requirements	Volume Requirements	Size Requirements	Data Location	Readiness Status

EXPECTED RESULTS:

	V	680	1	Verify that the operations in S-1 result in the export of the following: a) 25,000 HTTP PUT requests containing full granule metadata b) 25,000 HTTP DELETE requests sent to URLs reflecting granule resources, but containing no request body.		
	V	680	2	Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		
	V	680	3	Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		

1.1.3 Performance - QA Updates (ECS-ECSTC-566)

DESCRIPTION:

	S	690	1	**[Performance - QA Updates]** Perform the following operations, ensuring that different granules are used for each: a) 100,000 QA Updates		

PRECONDITIONS:

STEPS:

#	Action	Expected Result	Notes
1	Pre-Conditions		#comment
2	Setup		#comment
3	Configure BMGT to export either to a fake ECHO endpoint, or to a proxy which will allow logging of all exports.		
4	Ensure that the BMGT system is quiesced, with no requests in the PENDING state.		
5	Pause the dispatcher using the GUI		
6	Stop the automatic driver: EcBmBMGTAutoStop <MODE>		
7	S-1 Perform the following operations, ensuring that different granules are used for each: a) 100,000 QA Updates		#comment
8	Find 100,000 distinct granules in the same collection		
9	Add the granuleids to a file QA_GRANULEID_FILE From a command prompt: > psql -h p4dbl03 -U readonly -d ecs -c "SELECT GranuleId FROM aim.amgranule WHERE VersionId = <version id> AND DeleteFromArchive = 'N' AND DeleteEffectiveDate IS NULL AND ShortName = '<shortname>' ORDER BY GranuleId desc LIMIT 100000" -t -q | grep -E -o "[0-9]+" > QA_GRANULEID_FILE Where <shortname> and <version id> are the collection for which we are selecting granules. e.g. MOD10A1 and 86		
10	Script to find is in /tools/common/test/new_scripts/BMGTperf/GenerateQAUU_performance.sh. I used MOD10A1.086 because it is a collection in OPS mode with > 400K granules. Result file is in /usr/ecs/OPS/CUSTOM/data/DSS/QAUU/OPS_LaRC_QAUU.20130701194018 on p5dpl01. Flag was set to "Suspect".		#comment
11	Record the time as QA_START_TIME		
12	See the QAUU 609 Section 4.8.9: http://edhs1.gsfc.nasa.gov/waisdata/eed/pdf/609eed001_Rev01_File_7.pdf		#comment
13	Select the metadata file for one granule: Using one granule ID from the QA_GRANULEID_FILE, from a database command prompt: > SELECT p.path || '/' || f.archivemetfilename FROM ammetadatafile f JOIN dsmdxmlpath p ON f.archivepathid = p.archivepathid WHERE f.granuleid = <granule ID> Note the resulting file name.		

#	Action	Expected Result	Notes
14	Find the name of a measured parameter to update: > xpath <file name> '//MeasuredParameterContainer/ParameterName/text()' where <file name> is the previously noted xml file name. If more than one parameter name ('NODE' in xml terms) is returned, pick one (e.g. 'Day_Tile_Snow_Cover').		
15	Find the measured parameter's science quality flag value: > xpath <file name> "//MeasuredParameterContainer[ParameterName='<parameter name>']/QAFlags/ScienceQualityFlag/text()" where <parameter name> is the previously selected parameter name. Note the current flag value (e.g. 'Suspect')		
16	Select a different science quality flag value from the list of valid values in the properties file: > grep VALID_SCIENCE_QA_FLAGS /usr/ecs/${MODE}/CUSTOM/cfg/EcDsAmQaUpdateUtility.properties e.g. 'Passed'		
17	Create a QA update request file named according to the the QAUU 609: <MODE>_<Site>_QAUU.<YYYY><MM><DD><hh><mm><ss> e.g. > touch OPS_LaRC_QAUU.20130522231600		
18	Write a QA update request to update a science flag, using a GranuleUR (replace <TAB> with an actual tab character): First, add the file header: > echo "From GSFC" >> <request file> > echo "begin QAMetadataUpdate Science GranuleUR" >> <request file> where <request file> is the previously selected name for the QA update request file.		
19	Add the file contents: > cat QA_GRANULEID_FILE | awk '{print "${shortname}<TAB>${versionid}<TAB>${shortname}.${versionid}:" $1 "<TAB>${measured_parameter_name}<TAB>${qa_flag_value}<TAB>${qa_flag_explanation}" >> OPS_LaRC_QAUU.20130522231600 > cat QA_GRANULEID_FILE | awk '{printf "%1$s\t86\tSC:%1$s.%2$03d:%3$d\tALL\t%4$s\t%5$s\n", "<shortname>", <version id>, $1, "<quality flag>", "BmgtPerfTest_<request file>";}' >		

#	Action	Expected Result	Notes
	OPS_LaRC_QAUU.20130522231600 where <quality flag> is the newly selected quality flag value. 		
20	Add the file footer: > echo "end QAMetadataUpdate" >> <request file>		
21	Copy the QA update request file to the QA request directory configured in the properties file: To find the request directory location, from a command prompt: > sed -n 's/QA_REQUEST_DIR *= *//p' /usr/ecs/<mode>/CUSTOM/cfg/EcDsAmQaUpdateUtility.properties		
22	Run QAUU, specifying the filename (but not the path) if there are other request files in the request directory: EcDsAmQAUUStart <mode> -file <request file> -noprompt -noExitOnError -skipRecovery		
23	Verify that the QA Updates resulted in events in the DsMdGrEventHistory table: From a database command prompt: SELECT count(1) FROM DsMdGREventHistory WHERE eventtime > {QA_START_TIME} AND eventType = 'GRQAUPDATE' This should show 100,000 events.		
24	Restart the Automatic driver: > EcBmBMGTAutoStart <mode> verifying that BMGT.AutoDriver.PollingFrequency is set greater than 100,000		
25	Verify that all the events have been added to the BMGT requests queue: select count(1) from bg_export_request where enqueuetime > {QA_START_TIME} and exportQueue = 'EVENT' and granuleId is not null This should return 100,000		
26	record the time as BMGT_START_TIME		
27	Resume the dispatcher via the GUI		
28	Verification		#comment
29	V-1 Verify that the operations in S-1 result in the export of the following: a) 100,000 HTTP PUT requests containing full granule metadata		#comment
30	Inspect the proxy or fake ECHO log to verify that 100,000 HTTP PUT requests were sent		
31	V-2 Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		#comment
32	TBD		
33	V-3 Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		#comment

#	Action	Expected Result	Notes
34	Verify that all 100,000 requests are moved to the SUCCESS state within 1 hour after the recorded dispatcher resumption time. select count(1) from bg_export_request where enqueuetime > {QA_START_TIME} and exportQueue = 'EVENT' and granuleId is not null and status = 'SUCCESS' and completionTime < ({BMGT_START_TIME} + interval '1 hour') should return 100,000		

TEST DATA:

Crit id	Crit ccr no	Test Data Description	Data Type Requirements	Metadata Requirements	Volume Requirements	Size Requirements	Data Location	Readiness Status

EXPECTED RESULTS:

	V	690	1	Verify that the operations in S-1 result in the export of the following: a) 100,000 HTTP PUT requests containing full granule metadata		
	V	690	2	Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		
	V	690	3	Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		

1.1.4 Performance - Datapool Inserts/Updates/Deletes (ECS-ECSTC-567)

DESCRIPTION:

	S	700	1	**[Performance - Datapool Inserts/Updates/Deletes]** Perform the following operations, ensuring that different granules are used for each: a) 40,000 Datapool granule publications b) 40,000 Datapool granule unpublications c) 20,000 Datapool granule updates - perform this by moving one or more collections with a total of 20,000 granules.		

PRECONDITIONS:

STEPS:

#	Action	Expected Result	Notes
1	Pre-Conditions		#comment
2	Configure BMGT to export either to a fake ECHO endpoint, or to a proxy which will allow logging of all exports.		
3	Ensure that the BMGT system is quiesced, with no requests in the PENDING state.		
4	Stop the automatic driver EcBmBMGTAutoStop <MODE>		
5			#comment
6	Setup		#comment
7	S-1 Perform the following operations, ensuring that different granules are used for each: a) 40,000 Datapool granule publications		#comment
8	40,000 non public granules from collection C1 and write their IDs to a file select granuleid from AmGranule where ShortName = ${C1.ShortName} and VersionId = ${VersionId} and coalesce(isOrderOnly,'B') in ('Y', 'H') limit 40000 Write results to a file		
9	Actually the following query should be used to find 40K non-public granules to publish: select granuleid from AmGranule where ShortName = ${C1.ShortName} and VersionId = ${VersionId} and coalesce(isOrderOnly,'B') in ('Y', 'H') and coalesce(DeleteFromArchive, 'N') = 'N' and DeleteEffectiveDate IS NULL and granuleid not in (select granuleid from DsMdGranuleRestriction) limit 40000;		#comment
10	The below step didn't necessarily publish all granules we found by the above query because some of the candidates to be published already had a later version of the same data in the data pool caused by a duplicate granule ingested.		#comment
11	publish the granules. EcDlPublishUtilityStart <MODE> -ecs -file <FileName>		
12	S-1 Perform the following operations, ensuring that different granules are used for each: b) 40,000 Datapool granule unpublications		#comment
13	select 40,000 public granules from collection C2 and write their IDs to a file select granuleid from AmGranule where ShortName = ${C1.ShortName} and VersionId = ${VersionId} and		

#	Action	Expected Result	Notes
	coalesce(isOrderOnly,'B') ='B' limit 40000 Write results to a file		
14	unpublish the granules. EcDlUnPublishUtilityStart -mode <MODE> -file <FileName>		
15	S-1 Perform the following operations, ensuring that different granules are used for each: c) 20,000 Datapool granule updates - perform this by moving one or more collections with a total of 20,000 granules.		#comment
16	Find a collection C3 with 20,000 public granules: Ensure: select count(1) from AmGranule where granuleId not in (select granuleid from DsMdGranuleRestriction) and shortname = ${C3.ShortName} and versionid = ${C3.VersionId}		
17	Move collection C3 to a different filesystem (e.g., from ${FS_OLD} to ${FS_NEW}): EcDlMoveCollection.pl ${MODE} -shortname ${SHORTNAME} -versionid ${VERSIONID} -sourcefs ${FS_OLD} -targetfs ${FS_NEW} -verbose		
18	We ended up using collection MYD13AQ1.086 and moved this collection of 29,021 granules from FS4 to FS2. This took		#comment
19	Note the current Time, $StartTime		
20	Start the auto driver EcBmBMGTAutoStart <MODE> (and ensure that the dispatcher is running by inspecting the GUI or log)		
21	Verification		#comment
22	V-1 Verify that the operations in S-1 result in the export of the following: a) 100,000 HTTP PUT requests containing full granule metadata		#comment
23	Inspect the proxy or fake ECHO log to verify that 100,000 HTTP PUT requests were sent		
24	V-2 Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		#comment
25	TBD		
26	V-3 Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		#comment
27	Verify that all exports are complete withing 1 hour of $StartTime		

TEST DATA:

Crit id	Crit ccr no	Test Data Description	Data Type Requirements	Metadata Requirements	Volume Requirements	Size Requirements	Data Location	Readiness Status

EXPECTED RESULTS:

	V	700	1	Verify that the operations in S-1 result in the export of the following: a) 100,000 HTTP PUT requests containing full granule metadata		
	V	700	2	Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		
	V	700	3	Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		

1.1.5 Performance - Granule Updates (ECS-ECSTC-568)

DESCRIPTION:

	S	710	1	**[Performance - Granule Updates]** Perform the following operations, ensuring that different granules are used for each: a) Use the XRU to replace the metadata for 5,000 granules. b) Restrict 5,000 granules. c) Unrestrict 5,000 granules. d) Hide 5,000 granules. e) Unhide 5,000 granules.		

PRECONDITIONS:

STEPS:

#	Action	Expected Result	Notes
1	Pre-Conditions		#comment
2	Configure BMGT to export either to a fake ECHO endpoint, or to a proxy which will allow logging of all exports.		

#	Action	Expected Result	Notes
3	Set BMGT.AutoDriver.MaxEvents to 60000 (instead of 300000) on BMGT GUI Configuration tab.		#comment
4	Ensure the collections are installed and enabled for collection and granule export. This can be confirmed on the BMGT GUI (Collection Configuration tab) or via the database as follows: SELECT granuleexportflag , collectionexportflag FROM bg_collection_configuration WHERE shortname = <SHORTNAME> AND versionid = <VERSIONID>; If they are not enabled, UPDATE bg_collection_configuration SET granuleexportflag = 'Y', collectionexportflag = 'Y' WHERE shortname = <SHORTNAME> AND versionid = <VERSIONID>; Allow any export requests generated by these actions to complete. 		
5	Restrict 5000 granules:		#comment
6	Ensure that there are enough granules in the collection by ensuring that this query returns a number higher than 5000: select count(1) from AmGranule where granuleId not in (select granuleid from DsMdGranuleRestriction) and shortname = ${C3.ShortName} and versionid = ${C3.VersionId}		
7	Find a restriction flag value to use. valid values are found by running: select * from DsMdRestrictionFlag where restrictionFlag != 255 note one of the RestrictionFlag column values and record it as ${RestrictionFlag} If no rows are returned, insert a new row with restrictionFlag value < 255 and > 0		
8	insert into DsMdGranuleRestriction (granuleid, restrictionflag, unpublishflag, lastupdate) values select granuleid, ${RestrictionFlag}, 'N', lastupdate) from AmGranule where granuleId not in (select granuleid from DsMdGranuleRestriction) and shortname = ${C3.ShortName} and versionid = ${C3.VersionId} limit 5000 		
9	Allow the automatic driver to run and the requests generated for these restrictions to complete		
10	Ensure that the BMGT system is quiesced, with no requests in the PENDING state.		
11	Pause the dispatcher using the GUI		
12	Setup		#comment
13	S-1 Perform the following operations, ensuring that different granules are used for each: a) Use the XRU to replace the metadata for 5,000 granules.		#comment

#	Action	Expected Result	Notes
14	Get metadata file paths for 5000 granules: select 'cp ' || p.path || '/' || f.archivemetfilename || ' ${stagingDirPath}' from AmGranule g join ammetadatafile f on f.granuleid = g.granuleid join DsMdXmlPath p on f.archivepathid = p.archivepathid where g.ShortName= {$C1.ShortName} and g.VersionId = ${C1.VersionId} limit 5000 Where ${StagingDirPath} is a path where metadata fiels will be copied and modified. direct the results fo this query to a ${ScriptFile}		
15	Actually to find the appropriate collection to modify XML, we used this query: select a.shortname, a.versionid, count(1) from amgranule a, bg_collection_configuration b where a.granuleId not in (select granuleid from DsMdGranuleRestriction) and a.shortname = b.shortname and a.versionid = b.versionid and b.granuleexportflag = 'Y' and b.collectionexportflag = 'Y' and a.deleteeffectivedate IS NULL and deletefromarchive = 'N' group by a.shortname, a.versionid order by count(1); This will avoid finding things that are marked for deletion.		#comment
16	Rewrote the above step as follows: o cpit.out; select 'cp ' || p.path || '/' || f.archivemetfilename || ' ${stagingDirPath}' from AmGranule g join ammetadatafile f on f.granuleid = g.granuleid join DsMdXmlPath p on f.archivepathid = p.archivepathid where g.ShortName= {$C1.ShortName} and g.VersionId = ${C1.VersionId}; Where ${StagingDirPath} is a path where metadata files will be copied and modified. The results will be in script file cpit.out. Edit cpit.out and remove the first two line and last two lines of the file.		#comment
17	Copy metadata files to the statging location: mkdir ${StagingDirPath} chmod 777 ${ScriptFile} ./${ScriptFile}		
18	Modify the metadata: In the ${StagingDirPath} directory, run sed -ie '/LocalVersionID/{s/>\(.*\)</>\1-MODIFIED</g}' *.xml		
19	Actually prior to the above step, you need to cd ${StagingDirPath}. Then the command to modify an xml file in place is to run: sed -i '/LocalVersionID/{s/>\(.*\)</>\1-MODIFIED</g}' *.xml You also need to ensure that the metadata you are modifying does indeed contain "LocalVersionID". This test was tried with AE_Land.002 and that did not have LocalVersionID. We used MYD10A1.086 and MYD11_L2.086.		#comment
20	Replace the metadata: on x4dpl01 run EcDsAmXruStart <MODE> -xmldir ${StagingDirPath}		
21	The above step should say "Replace the metadata: on x5dpl01 run		#comment

#	Action	Expected Result	Notes
	EcDsAmXruStart <MODE> -xmldir ${StagingDirPath}		
22	S-1 Perform the following operations, ensuring that different granules are used for each: b) Restrict 5,000 granules.		#comment
23	Ensure that there are enough granules in the collection by ensuring that this query returns a number higher than 5000: select count(1) from AmGranule where granuleId not in (select granuleid from DsMdGranuleRestriction) and shortname = ${C2.ShortName} and versionid = ${C2.VersionId}		
24	Find a restriction flag value to use. valid values are found by running: select * from DsMdRestrictionFlag where restrictionFlag != 255 note one of the RestrictionFlag column values and record it as ${RestrictionFlag} If no rows are rertuend, insert a new row with restrictionFlag value < 255 and > 0		
25	insert into DsMdGranuleRestriction (granuleid, restrictionflag, unpublishflag, lastupdate) values select granuleid, ${RestrictionFlag}, 'N', lastupdate) from AmGranule where granuleId not in (select granuleid from DsMdGranuleRestriction) and shortname = ${C2.ShortName} and versionid = ${C2.VersionId} limit 5000		
26	S-1 Perform the following operations, ensuring that different granules are used for each: c) Unrestrict 5,000 granules.		#comment
27	delete DsMdGranuleRestriction r join AmGranule g on r.granuleid = g.granuleid where g.shortname = ${C3.ShortName} and g.versionid = ${C3.VersionId}		
28	Actually this works: delete from DsMdGranuleRestriction where granuleid in (select granuleid from amgranule where shortname = 'AE_Land' and versionid = 2);		#comment
29	S-1 Perform the following operations, ensuring that different granules are used for each: d) Hide 5,000 granules.		#comment
30	Ensure that there are enough granules in the collection by ensuring that this query returns a number higher than 5000: select count(1) from AmGranule where DeleteFromArchive != 'H' and shortname = ${C4.ShortName} and versionid = ${C4.VersionId}		
31	update AmGranule set DeleteFromArchive = 'H' where g.shortname = ${C4.ShortName} and g.versionid = ${C4.VersionId} limit 5000		
32	S-1 Perform the following operations, ensuring that different granules are used for each: e) Unhide 5,000 granules.		#comment
33	Ensure that there are enough granules in the collection by ensuring that this		

#	Action	Expected Result	Notes
	query returns a number higher than 5000: select count(1) from AmGranule where DeleteFromArchive = 'H' and shortname = ${C5.ShortName} and versionid = ${C5.VersionId}		
34	update AmGranule set DeleteFromArchive = 'H' where g.shortname = ${C5.ShortName} and g.versionid = ${C5.VersionId} limit 5000		
35	Actually the above query should be: update AmGranule set DeleteFromArchive = 'N' where granuleid in (select granuleid from AmGranule where shortname = ${C5.ShortName} and g.versionid = ${C5.VersionId} limit 5000);		#comment
36	Resume BMGT dispatcher using the GUI, recording the time at which the resumption occurs		
37	Verification		#comment
38	V-1 Verify that the operations in S-1 result in the export of the following: a) 25,000 HTTP PUT requests containing full granule metadata		#comment
39	Inspect the proxy or fake ECHO log to verify that 25,000 HTTP PUT requests were sent		
40	V-2 Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		#comment
41	TBD		
42	V-3 Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		#comment
43	*Verify that all 25,000 requests are moved to the SUCCESS state within 1 hour after the recorded dispatcher resumption time.		

TEST DATA:

Crit id	Crit ccr no	Test Data Description	Data Type Requirements	Metadata Requirements	Volume Requirements	Size Requirements	Data Location	Readiness Status

EXPECTED RESULTS:

	V	710	1	Verify that the operations in S-1 result in the export of the following: a) 25,000 HTTP PUT requests containing full granule metadata		

	V	710	2	Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		
	V	710	3	Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		

1.1.6 Performance - Collection Inserts/Updates (ECS-ECSTC-569)

DESCRIPTION:

	S	720	1	**[Performance - Collection Inserts/Updates]** Perform the following operations, ensuring that different collections are used for each: a) Insert or update 900 collections. If 900 collections do not exist, then insert or update all of the available collections.		

PRECONDITIONS:

STEPS:

#	Action	Expected Result	Notes
1	Pre-Conditions		#comment
2	Configure BMGT to export either to a fake ECHO endpoint, or to a proxy which will allow logging of all exports.		
3	Ensure the collections are installed and enabled for collection and granule export. This can be confirmed on the BMGT GUI (Collection Configuration tab) or via the database as follows: SELECT granuleexportflag , collectionexportflag FROM bg_collection_configuration WHERE shortname = <SHORTNAME> AND versionid = <VERSIONID>; If they are not enabled, UPDATE bg_collection_configuration SET granuleexportflag = 'Y', collectionexportflag = 'Y' WHERE shortname = <SHORTNAME> AND versionid = <VERSIONID>; Allow any export requests generated by these actions to complete.		
4	Ensure that the BMGT system is quiesced, with no requests in the PENDING state.		
5	Pause the dispatcher using the GUI		

#	Action	Expected Result	Notes
6	Setup		#comment
7	S-1 Perform the following operations, ensuring that different collections are used for each: a) Insert or update 900 collections. If 900 collections do not exist, then insert or update all of the available collections.		#comment
8	Update the DsGeESDTConfiguredType table manually to simulate collection updates by executing the following queries sequentially: update DsGeESDTConfiguredType set esdtstate = 'updating'; update DsGeESDTConfiguredType set esdtstate = 'installed';		
9	Actually the above queries should be: update DsGeESDTConfiguredType set esdtstate = 'updating' where configuredName in (select configuredName from DsGeESDTConfiguredType limit 900); & update DsGeESDTConfiguredType set esdtstate = 'installed' where esdtstate = 'updating';		#comment
10	Verify the dsmdgreventhistory table has a CLUPDATE event for each of the updated collections: select count(1) from dsmdgreventhistory where eventType = 'CLUPDATE' and eventTime > '${StartTime};		
11	Resume BMGT dispatcher using the GUI, recording the time at which the resumption occurs.		
12	Record the time at which all the requests were completed.		
13	Verification		#comment
14	V-1 Verify that the operations in S-1 result in the export of the following: a) 900 HTTP PUT requests containing full collection metadata. If there were not 900 collections to export, verify that the number of HTTP PUT requests matches the number of collections which were inserted/updated.		#comment
15	Inspect the proxy or fake ECHO log to verify that one HTTP PUT requests was sent per collection updated.		
16	V-2 Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		#comment
17	TBD		
18	V-3 Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		#comment
19	Verify that all requests are moved to the SUCCESS state within 1 hour after the recorded dispatcher resumption time.		
20	V-4 Verify that on average, the collection metadata was generated and		#comment

#	Action	Expected Result	Notes
	exported at a rate greater than or equal to 7 collections per second. Note: this equates to a rate of 25,000 per hour.</i>		
21	select count(1) as "Total complete requests", count(1) / (date_part('epoch', '${STOP_TIME}) - date_part('epoch', TIMESTAMP '${START_TIME}')) as "Average requests per second" from bg_export_request where enqueueTime > '${START_TIME}' and itemType = 'CL' and status = 'SUCCESS'; Ensure that the 'average requests per second' is greater than 7		

TEST DATA:

Crit id	Crit ccr no	Test Data Description	Data Type Requirements	Metadata Requirements	Volume Requirements	Size Requirements	Data Location	Readiness Status

EXPECTED RESULTS:

	V	720	1	Verify that the operations in S-1 result in the export of the following: a) 900 HTTP PUT requests containing full collection metadata. If there were not 900 collections to export, verify that the number of HTTP PUT requests matches the number of collections which were inserted/updated.		
	V	720	2	Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		
	V	720	3	Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		
	V	720	4	Verify that on average, the collection metadata was generated and exported at a rate greater than or equal to 7 collections per second. Note: this equates to a rate of 25,000 per hour.		

1.1.7 Performance - MISBR (ECS-ECSTC-570)

DESCRIPTION:

	S	730	1	**[Performance – MISBR]** Perform the following		

				operations, ensuring that different granules are used for each: a) Insert 4,300 MISR Level 1 granules, which are related to MISBR granules. b) Insert 4,300 MISR Level 2 granules, which are related to MISBR granules. c) Insert 860 MISBR granules which are related to MISR Level 1 and 2 science granules. d) Delete 4,300 MISR Level 1 granules, which are related to MISBR granules. e) Delete 4,300 MISR Level 2 granules, which are related to MISBR granules. f) Delete 860 MISBR granules which are related to MISR Level 1 and 2 science granules.		

PRECONDITIONS:

STEPS:

#	Action	Expected Result	Notes
1	Pre-Conditions		#comment
2	Ensure the test MISBR, MISR Level 1, MISR Level 2, and MIB2GEOP collections are installed.		
3	Ensure the test collections are configured to be public on ingest.		
4	Ensure test collections are configured for collection and granule export.		
5	Ensure AIM has no MISR granules that match the test MISBR granules. Use the xpath utility to get the Path Number, Orbit Number, Product Version, and Camera ID from the browse metadata file: xpath "//PSAs/PSA[PSAName='SP_AM_PATH_NO']/PSAValue/text()" misbr.xml xpath '//OrbitCalculatedSpatialDomain/OrbitCalculatedSpatialDomainContainer/OrbitNumber/text()' misbr.xml xpath "//PSAs/PSA[PSAName='SP_AM_MISR_ProductVersion']/PSAValue/text()" misbr.xml xpath '//Platform/Instrument/Sensor/SensorShortName/text()' misbr.xml Query AIM for matching MISR Level 1 granules: select esdt(g.shortname, g.versionid) ESDT, g.granuleid from amgranule g join dsmdmisrattributes a on g.granuleid = a.granuleid join dsmdmisrcamera c on g.granuleid = c.granuleid join		

#	Action	Expected Result	Notes
	dsmdmisrprocessingcriteria p on (g.shortname = p.shortname and g.versionid = p.versionid) where p.type = '1' and a.orbitnumber = $ORBITNUMBER and a.pathno = $PATHNUMBER and a.productversion = $PRODUCTVERSION and c.cameraid = $CAMERAID Query AIM for matching MISR Level 2 granules: select esdt(g.shortname, g.versionid) ESDT, g.granuleid from amgranule g join dsmdmisrattributes a on g.granuleid = a.granuleid join dsmdmisrcamera c on g.granuleid = c.granuleid join dsmdmisrprocessingcriteria p on (g.shortname = p.shortname and g.versionid = p.versionid) where p.type = '2' and a.orbitnumber = $ORBITNUMBER and a.pathno = $PATHNUMBER and c.cameraid = 'AN'		
6	Ensure AIM has no MISBR granules that match the test MISR granules. Use the xpath utility to get the Path Number, Orbit Number, Product Version, and Camera ID from the granule metadata files: xpath "//PSAs/PSA[PSAName='SP_AM_PATH_NO']/PSAValue/text()" granule.xml xpath '//OrbitCalculatedSpatialDomain/OrbitCalculatedSpatialDomainContainer/OrbitNumber/text()' granule.xml xpath "//PSAs/PSA[PSAName='SP_AM_MISR_ProductVersion']/PSAValue/text()" xpath '//Platform/Instrument/Sensor/SensorShortName/text()' granule.xml Query AIM for MISBR matching the MISR Level 1 granules: select esdt(g.shortname, g.versionid) ESDT, g.granuleid from amgranule g join dsmdmisrattributes a on g.granuleid = a.granuleid join dsmdmisrcamera c on g.granuleid = c.granuleid join dsmdmisrprocessingcriteria p on (g.shortname = p.shortname and g.versionid = p.versionid) where p.type = 'B' and a.orbitnumber = $ORBITNUMBER and a.pathno = $PATHNUMBER and a.productversion = $PRODUCTVERSION and c.cameraid = $CAMERAID Query AIM for MISBR matching the MISR Level 2 granules: select esdt(g.shortname, g.versionid) ESDT, g.granuleid from amgranule g join dsmdmisrattributes a on g.granuleid = a.granuleid join dsmdmisrcamera c on g.granuleid = c.granuleid join dsmdmisrprocessingcriteria p on (g.shortname = p.shortname and g.versionid = p.versionid) where p.type = 'B' and a.orbitnumber = $ORBITNUMBER and a.pathno = $PATHNUMBER and c.cameraid = 'AN'		
7	Ensure EcDlInsertUtility.properties has this: <br		

#	Action	Expected Result	Notes
	/>MISR_SPECIAL_PROCESSING=Y Bounce DPAD after changing the value: ./EcDlActionDriverStart $MODE		
8	Ensure that a TCP proxy or mock ECHO is capturing BMGT traffic.		
9	Ensure the BMGT dispatcher and auto driver are running: ./EcBmBMGTAppStart $MODE	ps auxww | sed -n '/OPS\/CUSTOM/{;s/.* -D\(Bmgt[^]*\).*/\1/p;}' BmgtComponent=EcBmDispatcher BmgtComponent=EcBmAuto BmgtComponent=EcBmMonitor	
10	Actually we make sure that the Dispatcher is down at this point. Do not bring it up yet.		#comment
11	Setup		#comment
12	S-1 Perform the following operations, ensuring that different granules are used for each:		#comment
13	a) Insert 4,300 MISR Level 1 granules, which are related to MISBR granules.		#comment
14	Perform an ingest on 4,300 MISR L1 granules and their related MISBR		
15	b) Insert 4,300 MISR Level 2 granules, which are related to MISBR granules.		#comment
16	Perform an ingest on 4,300 MISR L2 granules and their related MISBR		
17	c) Insert 860 MISBR granules which are related to MISR Level 1 and 2 science granules.		#comment
18	Ensure that the above MISR science ingests contain at least 860 MISBR granules as well.		
19	d) Delete 4,300 MISR Level 1 granules, which are related to MISBR granules.		#comment
20	Perform a logical or physical delete on 4,300 MISR L1 granules and their related MISBR		
21	e) Delete 4,300 MISR Level 2 granules, which are related to MISBR granules.		#comment
22	Perform a logical or physical delete on 4,300 MISR L2 granules and their related MISBR		
23	f) Delete 860 MISBR granules which are related to MISR Level 1 and 2 science granules.		#comment
24	Ensure that the above MISR science deletes contain at least 860 MISBR granules as well.		

#	Action	Expected Result	Notes
25	Bring up AutoDriver and Dispatcher		#comment
26	Verification		#comment
27	V-1 Verify that the operations in S-1 result in the export of the following: a) 4,300 HTTP PUT requests containing full granule metadata for MISR Level 1 granules. b) 4,300 HTTP PUT requests containing full granule metadata for MISR Level 2 granules. c) 860 HTTP PUT requests containing full granule metadata for MISBR granules. a. In addition, 860 HTTP PUT requests containing the full granule metadata for the MISR science granules related to the inserted MISBR granules. d) 4,300 HTTP DELETE requests containing no body, but pointing to a URL which relates to a MISR level 1 granule. e) 4,300 HTTP DELETE requests containing no body, but pointing to a URL which relates to a MISR level 2 granule. f) 860 HTTP DELETE requests containing no body, but pointing to a URL which relates to a MISBR granule. a. In addition, 860 HTTP PUT requests containing the full granule metadata for the MISR science granules related to the deleted MISBR granules. For a total of 11,180 HTTP PUT requests, and 9.460 HTTP DELETE requests.		
28	V-2 Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		
29	V-3 Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		

TEST DATA:

Crit id	Crit ccr no	Test Data Description	Data Type Requirements	Metadata Requirements	Volume Requirements	Size Requirements	Data Location	Readiness Status

EXPECTED RESULTS:

	V	730	1	Verify that the operations in S-1 result in the export of the following: a) 4,300 HTTP PUT requests containing full granule metadata for MISR Level 1 granules. b) 4,300 HTTP PUT requests containing full granule metadata for MISR Level 2 granules. c) 860 HTTP PUT requests containing full granule metadata for MISBR granules. a. In addition, 860 HTTP PUT requests containing the full granule metadata for the MISR science granules related to the inserted MISBR granules.		

				d) 4,300 HTTP DELETE requests containing no body, but pointing to a URL which relates to a MISR level 1 granule. e) 4,300 HTTP DELETE requests containing no body, but pointing to a URL which relates to a MISR level 2 granule. f) 860 HTTP DELETE requests containing no body, but pointing to a URL which relates to a MISBR granule. a. In addition, 860 HTTP PUT requests containing the full granule metadata for the MISR science granules related to the deleted MISBR granules. For a total of 11,180 HTTP PUT requests, and 9.460 HTTP DELETE requests.		
	V	730	2	Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		
	V	730	3	Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		

1.1.8 Performance - Long Form Verification (ECS-ECSTC-571)

DESCRIPTION:

	S	740	1	**[Performance - Long Form Verification]** Perform a long form verification which contains 100,000 granules.		

PRECONDITIONS:

STEPS:

#	Action	Expected Result	Notes
1	Pre-Conditions		#comment
2	Configure BMGT to export either to a fake ECHO endpoint, or to a proxy which will allow logging of all exports.		
3	Ensure that the BMGT system is quiesced, with no requests in the PENDING state.		
4	Pause the BMGT dispatcher via the GUI		
5	Setup		#comment
6	S-1 Perform a long form verification which contains 100,000 granules.		#comment
7	Identify 100,000 granules whose collections have previously been enabled for		

#	Action	Expected Result	Notes
	granule export. Write the granule ids to a file.		
8	Perform a long form verification of these granules: EcBmBMGTManualStart <MODE> -metg -long -gf <File>		
9	Note the current time, $StartTime		
10	Resume the BMGT dispatcher via the GUI		
11	Verification		#comment
12	V-1 Verify that the operations in S-1 result in the export of 100,000 HTTP PUT requests containing full granule metadata.		#comment
13	Inspect the proxy or fake ECHO log to verify that 100,000 HTTP PUT requests were sent		
14	V-2 Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		#comment
15	TBD		
16	V-3 Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		#comment
17	Verify that all exports are complete withing 1 hour of $StartTime		

TEST DATA:

Crit id	Crit ccr no	Test Data Description	Data Type Requirements	Metadata Requirements	Volume Requirements	Size Requirements	Data Location	Readiness Status

EXPECTED RESULTS:

	V	740	1	Verify that the operations in S-1 result in the export of 100,000 HTTP PUT requests containing full granule metadata.		
	V	740	2	Inspect the database metrics to ensure that during the process of this test, the database suffers no more than a 10% performance degradation.		
	V	740	3	Verify that all exports resulting from S-1 complete within 1 hour of the start of the test.		