Spectra Tape Libraries

XML Command Reference

Notices

Except as expressly stated herein, Spectra Logic Corporation makes its products and associated documentation on an "AS IS" BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE, BOTH OF WHICH ARE EXPRESSLY DISCLAIMED. In no event shall Spectra Logic be liable for any loss of profits, loss of business, loss of use or data, interruption of business, or for indirect, special, incidental or consequential damages of any kind, even if Spectra Logic has been advised of the possibility of such damages arising from any defect or error.

Information furnished in this manual is believed to be accurate and reliable. However, no responsibility is assumed by Spectra Logic for its use. Due to continuing research and development, Spectra Logic may revise this publication from time to time without notice, and reserves the right to change any product specification at any time without notice.

Trademarks

BlackPearl, BlueScale, CC, RioBroker, Spectra, SpectraGuard, Spectra Logic, StorCycle, TeraPack, TFinity, and TranScale are registered trademarks of Spectra Logic Corporation. Eon Protect and SeeVault are trademarks of Spectra Logic Corporation. MigrationPass is a service mark of Spectra Logic Corporation. All rights reserved worldwide. All other trademarks and registered trademarks are the property of their respective owners.

Part Number

90940114 Revision S

Revision History

Revision	Date	Description
P	December 2019	Updated for BlueScale12.8.02. Added: ▪ libraryUpTimeSeconds, chassisCount, frameType, SDInfo, and freeSpaceInMB parameters to the libraryStatus.xml response. ▪ cleaningCyclesTotal and cleaningCyclesLeft parameters to the inventory.xml?action=list response. ▪ robotStatus parameter in the libraryStatus.xml?action=list response for T200, T380, and T680 libraries.
Q	August 2020	Added inventory.xml?action=move and inventory.xml?action=getMoveResults.
R	November 2020	Added Security traceType to traces.xml and added slotSource and slotOffset to inventory.xml?action=list.
S	June 2021	Added utils.xml?action=`displayRoboticsCommunicationStatistics` and `utils.xml?action=gatherRoboticsCommunicationStatistics`

Note: To make sure you have the most current version of this guide check the Spectra Logic support portal at support.spectralogic.com/documentation. To make sure you have the release notes for the most current version of the BlueScale software, log into the Spectra Logic Technical Support portal at support.spectralogic.com. The release notes contain updates to the this guide since the last time it was revised.

End User License Agreement

1. READ CAREFULLY

YOU SHOULD READ THE FOLLOWING TERMS AND CONDITIONS BEFORE ACCEPTING THIS END-USER LICENSE AGREEMENT ("EULA"). THIS EULA IS A LEGAL AGREEMENT BETWEEN YOUR ORGANIZATION, THE END USER, AND SPECTRA LOGIC CORPORATION ("SPECTRA") FOR THE SPECTRA SOFTWARE PRODUCT WHICH INCLUDES COMPUTER SOFTWARE AND MAY INCLUDE ASSOCIATED MEDIA, PRINTED MEDIA, AND "ONLINE" OR ELECTRONIC DOCUMENTATION (COLLECTIVELY, "SOFTWARE PRODUCT"). BY INSTALLING, COPYING, OR OTHERWISE USING THE SOFTWARE PRODUCT, YOU AGREE TO BE BOUND BY THE TERMS OF THIS EULA. IF YOU DO NOT AGREE TO THE TERMS OF THIS EULA, YOU MAY NOT INSTALL, COPY, DOWNLOAD OR USE THE SOFTWARE PRODUCT. YOU AGREE THAT YOUR USE OF THE SOFTWARE ACKNOWLEDGES THAT YOU HAVE READ THIS AGREEMENT, UNDERSTAND IT, AND AGREE TO BE BOUND BY ITS TERMS AND CONDITIONS.

2. OWNERSHIP

It is understood and agreed that Spectra Logic Corporation, a Delaware corporation with offices at 6285 Lookout Road, Boulder, CO 80301 ("Licensor") is the owner of all right, title and interest to the Software Product, regardless of the media or form of the original download, whether by the World Wide Web, disk or otherwise. You, as licensee ("Licensee") through your downloading, installing, copying or use of this product do not acquire any ownership rights to the Software Product.

3. GENERAL

The Software Product is licensed, not sold, to you by Spectra for use only under the terms of this EULA. The Software Product is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The rights granted herein are limited to Spectra's and its licensors' intellectual property rights in the Software Product and do not include any other patents or intellectual property rights. The terms of this EULA will govern any software upgrades provided by Spectra that replace and/or supplement the original Software Product, unless such upgrade is accompanied by a separate license in which case the terms of that license will govern.

4. SOFTWARE PRODUCT

The Software Product, as used in this EULA, means, collectively and/or as applicable:

- The Software Product package;
- Any and all contents, components, attachments, software, media, and code with which this Agreement is provided and delivered;
- Any and all images, photographs, art, art work, clip art, fonts or other artistic works (the "Art Work");
- Related explanatory written materials and instructions, and any other possible documentation related thereto ("Documentation"); and
- Upgrades, modified versions, updates, additions and copies of the Software Product (the "Upgrades"), if any, licensed to by Spectra under this EULA.

5. GRANT OF LICENSE AND RESTRICTIONS

A. Spectra grants you a non-exclusive, non-transferable End-User license right to install the Software Product solely for the purpose for which it was created.

B. Unless provided otherwise in the Documentation or by prior express written consent of Spectra, you shall not display, modify, reproduce and distribute any Art Work, or portion(s) thereof, included with or relating to the Software Product, if any. Any such authorized display, modification, reproduction and distribution shall be in full accord with this EULA. Under no circumstances will your use, display, modification, reproduction and distribution of the Art Work give you any Intellectual Property or Proprietary Rights of the Art Work. All rights, title, and interest belong solely to Spectra.

C. Except for the initial loading of the Software Product, you shall not, without Spectra's express written consent:

- Copy or reproduce the Software Product; or
- Modify, adapt, or create derivative works based on the Software Product or any accompanying materials.

6. DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS

A. Spectra will provide you with support services related to the Software Product ("Support"). Such Support will be provided in accordance with the Spectra Master Support Agreement, available for download and viewing on the Spectra Corporate Web site. Use of Support is governed by this EULA and Spectra's Master Support Agreement.

B. Any supplemental software, code, content, or media provided to you in the course of Support shall be considered part of the Software Product and subject to the terms and conditions of this EULA.

C. Spectra retains all right, title, and interest in and to the Software Product, and any rights not granted to you herein are reserved by Spectra. You hereby expressly agree not to extract information, reverse engineer, disassemble, decompile, or translate the Software Product, or otherwise attempt to derive the source code of the Software, except to the extent allowed under any applicable law. In the event that such activities are permitted by applicable law, any information you, or your authorized agent, discover shall be promptly disclosed to Spectra and shall be deemed the confidential information of Spectra.

D. You shall not modify, sublicense, assign, or transfer the Software Product or any rights under this EULA, except as expressly provided in this EULA. Any attempt to sublicense, assign, or transfer any of the rights, duties, or obligations will be void.

E. You may permanently transfer all of your rights under this EULA, provided you retain no copies. The other party must agree to accept the terms and conditions of the EULA.

7. ALL RESERVED

All rights not expressly granted herein are reserved by Spectra.

8. TERM

A. This License is effective until terminated. Licensee may terminate it at any time by destroying the Software Product with all copies, full or partial, and removing all of its component parts.

B. Your rights under this EULA will terminate automatically without notice from Spectra if you fail to comply with any term(s) or condition(s) of this EULA. In such event, no notice shall be required by Spectra to effect such termination.

C. Upon termination of this EULA, you shall cease all use of the Software Product and destroy all copies, full or partial, together with all backup copies, modifications, printed or written materials, and merged portions in any form and remove all component parts of the Software Product.

9. INTELLECTUAL PROPERTY RIGHTS

A. Spectra shall retain all right, title, and interest in the Software Product and to any modifications or improvements made thereto, and any upgrades, updates or Documentation provided to End User. End User will not obtain any rights in the Software Product, its updates, upgrades, and Documentation, as a result of its responsibilities hereunder.

B. B. End User acknowledges Spectra's exclusive rights in the Software Product and that the Software Product is unique and original to Spectra and that Spectra is owner thereof. Unless otherwise permitted by law, End User shall not, at any time during or after the effective Term of the Agreement, dispute or contest, directly or indirectly, Spectra's exclusive right and title to the Software Product or the validity thereof.

10. U.S. GOVERNMENT END USERS

The Software Product and related documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable. The Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other End Users pursuant to the terms and conditions herein. Unpublished rights reserved under the copyright laws of the United States.

11. EXPORT LAW ASSURANCES

You may not use or otherwise export or re-export the Software Product except as authorized by United States law and the laws of the jurisdiction in which the Software Product was obtained. In particular, but without limitation, the Software Product may not be exported or re-exported (a) into (or to a nation or resident of) any U.S. embargoed countries or (b) to anyone on the U.S. Treasury Department's list of Specially Designated Nationals or the U.S. Department of Commerce Denied Persons List or Entity List. By installing or using any component of the Software Product, you represent and warrant that you are not located in, under control of, or a national or resident of any such country or on any such list.

12. DISCLAIMER OF WARRANTIES

YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT USE OF THE SOFTWARE PRODUCT IS AT YOUR SOLE RISK AND THAT THE ENTIRE RISK AS TO SATISFACTORY QUALITY, PERFORMANCE, ACCURACY AND EFFORT IS WITH YOU. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, AND EXCEPT AS MAY BE STATED IN THE SPECTRA MASTER SERVICE AGREEMENT, THE SOFTWARE PRODUCT IS PROVIDED "AS IS," WITH ALL FAULTS AND WITHOUT WARRANTY OF ANY KIND, AND SPECTRA AND SPECTRA'S AFFILIATES (COLLECTIVELY REFERRED TO AS "SPECTRA" FOR THE PURPOSES OF SECTIONS 12 AND 13) HEREBY DISCLAIM ALL WARRANTIES AND CONDITIONS WITH RESPECT TO THE SOFTWARE PRODUCT, EITHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES AND/OR CONDITIONS OF MERCHANTABILITY, OF SATISFACTORY QUALITY, OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY, OF QUIET ENJOYMENT, AND NON-INFRINGEMENT OF THIRD-PARTY RIGHTS. SPECTRA DOES NOT WARRANT AGAINST INTERFERENCE WITH YOUR ENJOYMENT OF THE SOFTWARE PRODUCT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE PRODUCT WILL MEET YOUR REQUIREMENTS, THAT THE OPERATION OF THE SOFTWARE PRODUCT WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS IN THE SOFTWARE PRODUCT WILL BE CORRECTED. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY SPECTRA OR A SPECTRA AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES OR LIMITATION ON APPLICABLE STATUTORY RIGHTS OF A CONSUMER, SO THE ABOVE EXCLUSION AND LIMITATIONS MAY NOT APPLY TO YOU.

13. LIMITATION OF LIABILITY

TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL SPECTRA, ITS AFFILIATES OR LICENSEES, BE LIABLE FOR ANY SPECIAL, INCIDENTAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS) ARISING OUT OF THE USE OF OR INABILITY TO USE THE SOFTWARE PRODUCT OR THE PROVISION OF OR FAILURE TO PROVIDE SUPPORT SERVICES, EVEN IF SPECTRA HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. IN ANY CASE, SPECTRA'S ENTIRE LIABILITY UNDER ANY PROVISION OF THIS EULA SHALL BE LIMITED TO THE AMOUNT ACTUALLY PAID BY YOU FOR THE SOFTWARE PRODUCT; PROVIDED HOWEVER, IF YOU HAVE ENTERED INTO A MASTER SUPPORT AGREEMENT, SPECTRA'S ENTIRE LIABILITY REGARDING SUPPORT SERVICES SHALL BE GOVERNED BY THE TERMS OF THAT AGREEMENT. BECAUSE SOME STATES AND JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

14. CONTROLLING LAW AND SEVERABILITY

This EULA will be governed by and construed in accordance with the laws of the State of Colorado, as applied to agreements entered into and to be performed entirely within Colorado between Colorado residents. This EULA shall not be governed by the United Nations Convention on Contracts for the International Sale of Goods, the application of which is expressly excluded. If for any reason a court of competent jurisdiction finds any provision, or portion thereof, to be unenforceable, the remainder of this EULA shall continue in full force and effect.

Contacting Spectra Logic

To Obtain General Information

Spectra Logic Website: spectralogic.com

United States Headquarters

Spectra Logic Corporation
6285 Lookout Road
Boulder, CO 80301
USA
Phone: 1.800.833.1132 or 1.303.449.6400
International: 1.303.449.6400
Fax: 1.303.939.8844

European Office

Spectra Logic Europe Ltd.
329 Doncastle Road
Bracknell
Berks, RG12 8PE
United Kingdom
Phone: 44 (0) 870.112.2150
Fax: 44 (0) 870.112.2175

Spectra Logic Technical Support

Technical Support Portal: **support.spectralogic.com**

United States and Canada

Phone:
Toll free US and Canada: 1.800.227.4637
International: 1.303.449.0160

Europe, Middle East, Africa

Phone: 44 (0) 870.112.2185

Deutsch Sprechende Kunden

Phone: 49 (0) 6028.9796.507
Email: spectralogic@stortrec.de

Mexico, Central and South America, Asia, Australia, and New Zealand

Phone: 1.303.449.0160

Spectra Logic Sales

Website: shop.spectralogic.com

United States and Canada

Phone: 1.800.833.1132 or 1.303.449.6400
Fax: 1.303.939.8844
Email: sales@spectralogic.com

Europe

Phone: 44 (0) 870.112.2150
Fax: 44 (0) 870.112.2175
Email: eurosales@spectralogic.com

To Obtain Documentation

Spectra Logic Website: support.spectralogic.com/documentation

Contents

About This Guide

This reference describes using the XML command interface, which provides automation support for operating and monitoring of Spectra® Tape Series libraries (referred to as the *library*) using a set of XML commands instead of the BlueScale® user interface.

Note: The Spectra T50e library does not support the XML command interface.

Intended Audience

This command reference is intended for system administrators who are responsible for writing a programmatic interface for monitoring and operating the library without using the BlueScale user interface. The reference assumes a working knowledge of using a standard programming language such as Java®, Perl®, or Python®, as well as an understanding of standard XML command structure.

Related Information

For additional information about the Spectra Tape Series libraries and their drives, refer to the publications listed in this section.

Product Status

The Spectra Logic® Technical Support portal provides information about which products are currently supported and which are considered discontinued. To view information about discontinued products, log into the portal (see your library User Guide), open the Knowledge Base, and search using the term “discontinuance”.

Spectra Tape Series Libraries

This command reference and the following documents related to the Spectra Tape Series libraries are available as PDF files on the Spectra Logic website at support.spectralogic.com/documentation.

- The *User Guide* for each library describes configuring, operating, troubleshooting, and maintaining the library and its drives.
- The *Quick Reference Guide* for each library provides a quick reference for the user interface and instructions for performing day-to-day library operations such as powering on and off, and preparing, importing, and exporting media.
- The *Release Notes and Documentation Updates* for each library provide the most up-to-date information about the library, drives, and media.

  Note: The release notes are only available on the Spectra Logic Support portal.
- The *Spectra BlueScale Vision Camera User Guide* provides detailed information about installing and using the white BlueScale Vision Camera and software.
- The *Vivotek FD8361 Fixed Dome Network Camera User's Manual* provides detailed information about installing and using the black BlueScale Vision Camera and software.
- The *Spectra Encryption User Guide* provides detailed information about using BlueScale Encryption Standard and Professional Edition and the Spectra SKLM Encryption key management system. It also provides useful information about encryption best practices and recycling encrypted media.
- The *Spectra Tape Libraries SCSI Developer's Guide* provides detailed information about the SCSI and Fibre Channel commands used in the library.
- The *Spectra Tape Libraries Warnings* document provides all of the warnings found in Spectra Tape Series libraries documentation, in English and 27 other languages.

Spectra SKLM Server

For additional information that can assist you during the installation and configuration of your server, see the following website:
IBM Security Key Lifecycle Manager welcome page

KMIP

See the documentation specific to your server.

LTO Ultrium Tape Drives

The following documents provide information that is applicable to all IBM LTO tape drives.

- *IBM Tape Device Drivers Installation and User's Guide*

  Note: This guide also provides information about using the IBM Tape Diagnostic Tool (ITDT) to troubleshoot drive problems.

- *IBM TotalStorage LTO Ultrium Tape Drive: SCSI Reference* (LTO-1 through LTO-4)
- *IBM TotalStorage LTO Ultrium Tape Drive: SCSI Reference* (LTO-5 - LTO-7)

For drive-specific information, search for the product name (for example, LTO 5) on the documentation page on the IBM website. You can also search the IBM Support Portal at: ibm.com/support/entry/portal/Documentation.

TS11*xx* Technology Drives

The following documents provide information that is applicable to TS11*xx* technology drives.

- *IBM System Storage Tape Drive 3592 SCSI Reference*
- *IBM Tape Device Drivers Installation and User's Guide*

  Note: This guide also provides information about using the IBM Tape Diagnostic Tool (ITDT) to troubleshoot drive problems.

StorageTek T10000 Drives

The following documents provide information that is applicable to StorageTek T10000 drives.

- *StorageTek T10000 Tape Drive Operator's Guide*
- *Using StorageTek T10K Drives in a Spectra TFinity Library*

CHAPTER 1

Overview

This chapter provides an overview of the XML command interface for Spectra tape libraries and a description of how this reference is organized.

USING THE XML COMMAND INTERFACE

The XML command interface provides automation support for operating and monitoring the library using a set of XML commands instead of the BlueScale user interface. The following sections describe the general requirements and process for using the XML command interface.

Connectivity Requirements

The XML command interface requires an active Ethernet connection to the library's Library Control Module (LCM). This connection is the same one used to access the BlueScale web interface. The default port is 80. See "Configure Network Settings" in your library *User Guide* for information and instructions.

Issuing Commands

A standard programming language such as Java, Perl, or Python can be used to issue a series of XML commands to the library. In addition to issuing the XML commands, the programs can parse the XML-formatted data that the library returns as the command response and interpret any output generated by the command.

Command Syntax

All of the XML commands use standard web URL structure. The general syntax for an XML command is:

```
[IP Address]/gf/[base URL]?[parameter 1]=[value]&
   [parameter 2]=[value]&...&[parameter n]=[value]
```

where:

- `[IP Address]` = The IP address of the library
- `gf` = The name of the directory where the webserver used for command processing resides
- `[base URL]` = The base URL of the XML command
- `[parameter 1]` through `[parameter n]` = Parameters whose values further define how the library responds to the base URL command. These parameters are described in the following table.

Important The parameter names and their values are case-sensitive; most use "camelCase." If you fail to receive a response back from the library, make sure that you typed the action name and any parameter names and values correctly.

Important If you include an unsupported parameter value in the command, the command fails and returns an error message indicating that the command received was not a valid command for the library.

Important Parameters should be URL encoded. For example, 'Partition 1' should be 'Partition+1'.

Notes:
- In some cases, the base URL can be used without any additional parameters.
- The syntax statements in this guide show separators in **bold** to make them easier to see.

This parameter...	Specifies...
parameter 1=[*value*]	The name of the first parameter following the base URL and its value. **Important:** The first parameter must be separated from the base URL by a question mark (**?**) and from any additional parameters by an ampersand (**&**). **Notes:** ▪ There are a few scenarios where a parameter does not have a value. In these cases, the usage of the parameter is all that is required. For example, the parameter `autofinish` in the following command does not have a value: `package.xml?action=update&package=[package name]&autoFinish.` ▪ There are a few scenarios where a parameter can be blank. In these cases, there is nothing after the equal sign. For example, if no password is set for the su username, you can use the following command to log in: `login.xml?username=su&password=.`
parameter 2=[*value*] through **parameter *n*=[*value*]**	The name and value of each additional parameter for the command. Multiple parameters must be separated by an ampersand (**&**). **Note:** Not all commands include additional parameters.

Command Responses

With few exceptions, which are noted in the command descriptions, the library returns XML-formatted data in response to the XML command. The root tag for the XML command distinguishes the type of response:

- `<syntaxError>` indicates a Syntax Error, see Syntax Error Response on this page.
- `<error>` indicates a system error, see System Errors on page 19.
- `<progress>` indicates the results of a command sent with the progress parameter. See Progress for Extended Action Commands on page 19.
- *`<base URL>`* indicates that the command completed. The detailed description of the response from each action is given in the "Command Response" section for each action.

Syntax Error Response

If a command is entered incorrectly, the library returns a syntax error response. If "action=" is required but not included, or is mis-typed, the response includes a list of actions associated with the base url. For example, if you send the base url `autosupport.xml` without an action parameter, you receive the following response:

```
<syntaxError>
   <message>Invalid Syntax</message>
   <usage>
      <line>autosupport.xml</line>
      <line>Query string:</line>
      <line>action=<generateASL|getASLNames|getASL></line>
   </usage>
</syntaxError>
```

If the action parameter is correct, but another required parameter is missing, the response includes a list of required parameters and possible values. For example, if you send the command `optionskeys.xml?action=add`, but do not include a key to add, you will receive the following syntax error:

```
<syntaxError>
   <message>Add command must contain a key</message>
   <usage>
      <line>optionkeys.xml</line>
      <line>Query string:</line>
      <line>action=<add|list></line>
      <line>key=<key></line>
   </usage>
</syntaxError>
```

System Errors

If the web server encounters an error while processing a request or command, it returns the following XML-formatted data:

```
<error>
   <message>[message]</message>
   <description>[description]</description>
</error>
```

where the value for:

This parameter...	Indicates...
message	The error message posted by the web server.
description	Any additional details included in the message.

Progress for Extended Action Commands

When using the BlueScale user interface, certain commands that do not complete immediately bring the user to a progress page while the command processes. Once the command completes, the BlueScale page refreshes to show the results. This type of behavior is not possible with an XML interface.

To view the progress of an extended action initiated through the XML command interface, send the base URL command again with the **progress** parameter. The XML session that issued the original command must request progress until it receives a status response of **OK** or **FAILED** in order to be able to send other extended action commands.

Important

An extended action prevents other extended actions requests from the same session until the result of the original action is obtained via a progress command. Extended actions submitted from different sessions are queued and executed in the order received.

Notes:
- Commands that do not take a long time return any data associated with the command in the command response.
- To view a list of the extended actions and background operations that the library is processing, use the **taskList.xml** command (see taskList.xml on page 189).

Syntax `[command].xml?progress`

where `[command]` is the base URL of the command that was issued.

Command Response The command response depends on whether or not the command that was sent is still in progress, or whether another session is currently running an extended command.

When the command from the current session is in progress or complete

```
<progress>
   <activePage>
      [command name]
   </activePage>
   <message>[message]</message>
   <status>
      OK|FAILED|QUEUE|SUBMITTED|ACTIVE
   </status>
   <extraInformation>[string]</extraInformation>
</progress>
```

- `<activePage>` Displays the base URL for the command for which progress was requested.
- `<message>` Provides extra detail about the command status.
- `<status>` definitions:
  - **SUBMITTED** - The command was received. The command is in this state for a very short time.
  - **QUEUE** - The action is waiting to run. The corresponding `<message>` indicates what action is currently running and from what IP address that action was started.
  - **ACTIVE** - The action is currently running.
  - **OK** - The action completed successfully. This status is returned only once for a successful action.
  - **FAILED** - The action failed. This status is returned only once for a failed action.
- `<extraInformation>` is an optional string added by the user to help identify extended commands.

When no command is in progress

```
<progress>
   <message>No Pending Actions</message>
   <status>OK</status>
</progress>
```

When an extended action is running and progress is requested for a different command

```
<error>
   <message>An action is currently in progress.</message>
   <description>[descriptionText]</description>
</error>
```

The `<description>` indicates the action that is currently running.

Note: Use the **taskList.xml** command (see taskList.xml on page 189) for more information about currently running or queued commands.

Example Command and Response The following command:

```
autosupport.xml?progress
```

returns the following while the library is collecting an AutoSupport log:

```
<progress>
   <activePage>autosupport.xml</activePage>
   <message>Create AutoSupport Trouble Ticket Progress :
Gathering
      trouble ticket data files.</message>
   <status>ACTIVE</status>
</progress>
```

Library Login and Logout

In order to protect the library security, you must log into the library before you can issue any additional commands, just as you would when using the BlueScale software from the operator panel or the web interface (RLC).

Important The connection to the library is automatically closed after the idle time specified through the BlueScale user interface System Setup screen (see "Auto Logout Timeout" in your library *User Guide*) or can be closed by issuing a logout command.

Important Connections to the library through the XML command interface are included in the maximum of eight simultaneous remote sessions supported by the library.

Notes:
- The library must complete its initialization process before it will accept a login command.
- For full syntax information on the login command, see login.xml on page 98.
- For full syntax information on the logout command, see logout.xml on page 100.

Login Use the following XML command to log into the library:

```
[IP Address]/gf/login.xml?username=[username]&password=[password]
```

where `[IP Address]` specifies the IP address of the library and:

This parameter...	Specifies...
username	A valid username assigned to the library. The username is case sensitive. **Notes:** ▪ The specified user must have either superuser or administrator privileges in order to perform configuration operations. See "Understanding User Groups and Security" in your library *User Guide* for more information. ▪ Users assigned to the Operator group can move, import, and export media, but cannot access the more sensitive library operations such as configuration, diagnostics, and security.
password	The password associated with the username. ▪ If no password is set, this parameter is optional (`login.xml?username=su`) or can be blank (`login.xml?username=su&password=`).

The login command also returns a session ID cookie that is required for subsequent XML commands issued from the same location.

Important If you are using a web browser to send commands, the cookie is handled transparently by the browser. If you are sending XML commands using a scripting language, the script must include commands to manage the session.

Logout Use the following XML command to log out of the library:

`[IP Address]/gf/logout.xml`

where `[IP Address]` specifies the IP address of the library.

USING THIS COMMAND REFERENCE

The following chapters describe each of the available XML commands supported by the tape libraries. Each XML command has a unique base URL that includes the library IP address and the command. In addition, each base URL command may include parameters that configure specific actions to be performed.

To simplify locating information, this reference is organized alphabetically, first by the base URL for the command and then by the command variant as determined by the first parameter that follows the base URL. Both the base URL and each of the variants are referred to as commands.

- If a base URL can be used without any parameters, this command variant is described in the first section of the chapter for the base URL and is the identified by **[no parameters]** in the command reference heading.
- If the first parameter following the base URL is **action**, the value of the action parameter is used to identify the command variant.

For each command variant of the base URL, the chapter provides the command syntax and definitions of all variables and parameters. It also provides an example of the command usage and the response. Where appropriate, the chapter also includes a command sequence that provides an example of how a sequence of XML commands (and possibly operator actions at the physical library) are used to perform a series of related operations.

Note: Not all tape libraries support all of the command variables and parameters. In addition, some commands are not supported by all libraries. When a command or parameter is library-specific, that information is included in the description of the command.

This command reference uses the following conventions for describing the syntax and command response for each command:

- All XML command base URLs begin with the following string: `[IP Address]/gf/`. For clarity, this string is not included in the syntax statement or the example for each command.
- Depending on the browser or programming language you are using, you may need to precede the XML command with `http://` or, if SSL is enabled for the library, `https://`.
- Variables in the command syntax are shown as `[variable]`. **Do not include the bracket characters ([ ]) when you type variables.**
- The response for a command is formatted using open and close XML tags `<tag name>` and `</tag name>`, where each *tag name* corresponds to the name of the parameter for which data is returned in the command responses. In some cases, the XML tags delineate a group of related tags.
- The formatting of the command response depends on the output device. For clarity, this command reference indents each hierarchical level of the XML tags.

CHAPTER 2

autosupport

autosupport.xml

Use the **autosupport.xml** command to generate a new AutoSupport Log (ASL) file or retrieve a previously generated ASL file. ASL files contain logs of specific library operations. Technical Support uses ASL files during troubleshooting.

Action	
generateASL	this page
getASLNames	page 25
getASL	page 26

Note: See the "AutoSupport" chapter in your library *User Guide* for detailed information about configuring and using AutoSupport.

generateASL

Description Generates a new ASL file.

Syntax `autosupport.xml?action=generateASL [&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<autosupport>
   <status>OK</status>
   <message>[message text]</message>
</autosupport>
```

Progress Use `autoSupport.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
autosupport.xml?action=generateASL
```

immediately returns the following XML-formatted data:

```
<autosupport>
   <status>OK</status>
   <message>Started ASL creation. Set progress in your query for
      status.</message>
</autosupport>
```

When ASL file generation is complete, the response to `autosupport.xml?progress` contains `<status>OK</status>`. Use `autosupport.xml?action=getASLNames` to display the names of all ASL files, and then use `autoSupport.xml?action=getASL&name=[ASLName]` to retrieve the file.

getASLNames

Description Returns a list of the ASL files currently stored on the library.

Syntax `autosupport.xml?action=getASLNames`

Command Response The command immediately returns a list of ASL files currently stored on the library. The list of ASL files is returned using the following format:

```
<autosupport>
   <ASLNames>
      <ASLName>[HardwareID] [date] [time].asl</ASLName>
...
      <ASLName>[HardwareID] [date] [time].asl</ASLName>
   </ASLNames>
</autosupport>
```

where the value for:

This parameter...	Indicates...
ASLName	The filename of the ASL file, where the filename includes the value for each of the following variables: ▪ *HardwareID* = The library's hardware ID (serial number). ▪ *date* = The month, day, and year (mm-dd-yyyy) when the library generated the ASL file. ▪ *time* = The time (hh.mm.ss), based on a 24-hour clock, when the library generated the ASL file.

Example Command and Response The following command:

```
autosupport.xml?action=getASLNames
```

immediately returns the following XML-formatted data:

```
<autosupport>
   <ASLNames>
      <ASLName>0919402 01-13-2014 17.03.50.asl</ASLName>
      <ASLName>0919402 01-23-2014 12.17.21.asl</ASLName>
   </ASLNames>
</autosupport>
```

getASL

Description Retrieves the specified ASL file from the library.

Syntax `autosupport.xml?action=getASL&name=[ASLName]`

where the value for:

This parameter...	Specifies...
name	The name of the ASL file to retrieve.

Command Response The command immediately returns a stream of data containing the ASL file.

Example Command and Response The following command:

```
autosupport.xml?action=getASL&name=0919402 01-13-2014 17.03.50.asl
```

returns a file named `autosupport.zip` which contains the ASL data.

CHAPTER 3

controllers

controllers.xml

The **controllers.xml** command returns status information for the library controllers, Robotics Interface Modules (RIMs) and Fibre Channel Quad Interface Processors (F-QIPs), and enables or disables controller failover.

Notes:
- Unless otherwise specified, the features of both a RIM and RIM2 are the same and "RIM" is used to refer to both.
- This command was added in BlueScale12.6.41.
- This command is not supported for T120 libraries.

Action	
disableFailover	below
enableFailover	page 29
list	page 30

disable Failover

Description Disables controller redundancy for the specified primary controller.

Note: This action was added with BlueScale12.6.45.

Syntax `controllers.xml?action=disableFailover &controller=FR[integer]/DBA[integer]/F-QIP[integer] [&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
controller	The component identifier for the exporting RIM or F-QIP (primary controller) using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the controller. Not used in T120 library component identifiers. ▪ **F-QIP[*integer*]** = The designator of the controller bay where the QIP is installed. For all libraries except the T120, the value of [***integer***] is always **1**. For the T120 library, the value of [***integer***] is either **1** or **2**. **Note:** See "Component Identifiers" in your library *User Guide* for more information, including ranges for each part of the identifier.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<controllers>
   <disableFailover>
      <status>OK</status>
      <message>[message text]</message>
   </disableFailover>
</controllers>
```

Progress Use `controllers.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
controllers.xml?action=disableFailover&controller=FR1/DBA6/F-QIP1
```

immediately returns the following XML-formatted data:

```
<controllers>
   <disableFailover>
      <status>OK</status>
      <message>Started disableFailover. Set progress in your query
         for status.</message>
   </disableFailover>
</controllers>
```

and disables failover for the controller with the component identifier FR1/DBA6/F-QIP1. When the command is complete, the response to `controllers.xml?progress` contains `<status>OK</status>`.

enable Failover

Description Enables controller redundancy for the specified controller using the identified spare. Controller failover configures two RIMs or two F-QIPs as a failover pair. RIMs and F-QIPs can be mixed in a failover pair. A RIM2 can only be in a failover pair with another RIM2. One member of the pair (the primary controller) provides the robotic control path in the partition. The other controller is designated as the spare. In the event that the spare detects an internal problem with the primary controller, it automatically takes over all robotics control operations to provide uninterrupted operation.

Note: This action was added with BlueScale12.6.45.

Syntax `controllers.xml?action=enableFailover &controller=FR[integer]/DBA[integer]/ F-QIP[integer]&spare=FR[integer]/DBA[integer]/F-QIP[integer] [&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
controller	The component identifier for the exporting RIM or F-QIP using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **F-QIP[*integer*]** = The designator of the controller bay where the QIP is installed. For all libraries except the T120, the value of [***integer***] is always **1**. For the T120 library, the value of [***integer***] is either **1** or **2**. **Note:** See "Component Identifiers" in your library *User Guide* for more information, including ranges for each part of the identifier.
spare	The component identifier for the spare RIM or F-QIP using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]** as described above.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<controllers>
   <enableFailover>
      <status>OK</status>
      <message>[message text]</message>
   </enableFailover>
</controllers>
```

Progress Use `controllers.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
controllers.xml?action=enableFailover&controller=FR1/DBA5/F-
QIP1&spare=FR1/DBA6/F-QIP1
```

immediately returns the following XML-formatted data:

```
<controllers>
   <enableFailover>
      <status>OK</status>
      <message>Starting enableFailover. Set progress in
         your query for status.</message>
   </enableFailover>
</controllers>
```

and enables failover for the primary controller with the component identifier FR1/DBA5/F-QIP1 using the controller with the component identifier FR1/DBA6/F-QIP1 as the spare. When the command is complete, the response to `controllers.xml?progress` contains `<status>OK</status>`.

list

Description Returns controller status, type, firmware, failover configuration, and port configuration information for all controllers in the library.

Syntax `controllers.xml?action=list`

Command Response The command immediately returns the following XML-formatted data:

```
<controllers>
   <controller>
      <ID>FR[integer]/DBA[integer]/F-QIP[integer]</ID>
      <status>Normal|Missing|Impaired</status>
      <firmware>[value]</firmware>
      <type>
         8-Gbps FC RIM2|4-Gbps FC RIM|2-Gbps FC RIM|
         1-Gbps Ethernet RIM|4-Gbps Fibre Channel|
         2-Gbps Fibre Channel|1-Gbps Ethernet
      </type>
      <failoverFrom>
         FR[integer]/DBA[integer]/F-QIP[integer]
      </failoverFrom>
      <failoverTo>
         FR[integer]/DBA[integer]/F-QIP[integer]
      </failoverTo>
      <port>
         <name>A|B</name>
         <useSoftAddress>yes|no</useSoftAddress>
         <loopId>[value]</loopId>
         <initiatorEnabled>yes|no</initiatorEnabled>
         <fibreConnectionMode>
            loop|fabric|Auto-negotiate
         </fibreConnectionMode>
      </port>
      ...
   </controller>
   ...
<controllers>
```

where the value for:

This parameter...	Indicates...
ID	The component identifier for the exporting RIM or F-QIP (primary controller) using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **F-QIP[*integer*]** = The designator of the controller bay where the QIP is installed. For all libraries except the T120, the value of [***integer***] is always **1**. For the T120 library, the value of [***integer***] is either **1** or **2**. **Note:** See "Component Identifiers" in your library *User Guide* for more information, including ranges for each part of the identifier.
status	The operating status of the controller. Values: **Normal, Missing,** or **Impaired** (The controller is not responding on the library's internal network)
firmware	The firmware version in use by the controller.
type	The type of controller. Values: ▪ **8-Gbps FC RIM2** = RIM2 ▪ **4-Gbps FC RIM** = 4 Gb/second Fibre Channel RIM ▪ **2-Gbps FC RIM** = 2 Gb/second Fibre Channel RIM ▪ **1-Gbps Ethernet RIM** = 1 Gb/second Ethernet RIM ▪ **4-Gbps Fibre Channel** = 4 Gb/second F-QIP ▪ **2-Gbps Fibre Channel** = 2 Gb/second F-QIP ▪ **1-Gbps Ethernet** = 1 Gb/second E-QIP
failoverFrom	The component identifier of the **primary** controller configured in a failover pair, using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]** (see ID above). Only displays for the spare controller in a failover pair.
failoverTo	The component identifier of the **spare** controller configured in a failover pair, using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]** (see ID above). Only displays for the primary controller in a failover pair.
port	Information about how the controller port is configured. ▪ **name** - The designator for the port. Values: **A**, **B** ▪ **useSoftAddress** = Whether or not the port is configured to use soft addressing. Values: **yes**, **no** ▪ **loopId** - The configured fixed loop ID. Only displayed if **useSoftAddress** is **no**. ▪ **initiatorEnabled** - This is no longer used and will always show a value of **no.** ▪ **fibreConnectionMode** - The configured Fibre connection mode. Values: **Loop**, **Fabric**, **Auto-negotiate**

Example Command and Response The following command:

```
controllers.xml?action=list
```

immediately returns the following XML-formatted data:

```
<controllers>
   <controller>
      <ID>FR1/DBA6/F-QIP1</ID>
      <status>Normal</status>
      <firmware>8.15.42</firmware>
      <type>2-Gbps FC RIM</type>
      <port>
         <name>B</name>
         <useSoftAddress>yes</useSoftAddress>
         <loopId>0</loopId>
         <initiatorEnabled>no</initiatorEnabled>
         <fibreConnectionMode>Auto-negotiate</fibreConnectionMode>
      </port>
      <port>
         <name>A</name>
         <useSoftAddress>yes</useSoftAddress>
         <loopId>0</loopId>
         <initiatorEnabled>no</initiatorEnabled>
         <fibreConnectionMode>Auto-negotiate</fibreConnectionMode>
      </port>
   </controller>
</controllers>
```

CHAPTER 4

driveList

driveList.xml

Use the **driveList.xml** command to identify the drives in the library and perform troubleshooting operations on a specific drive.

Note: Refer to your library *User Guide* for detailed information about the requirements for using and replacing drives in the library.

[no parameters] or list

Description Returns detailed information about each drive in the library.

Syntax `driveList.xml?action=list`

Command Response The command returns the following XML-formatted data:

```
<driveList>
   <drive>
      <ID>[value]</ID>
      <driveStatus>OK|impaired|missing|unknown</driveStatus>
      <partition>[value]</partition>
      <partitionDriveNumber>[value]</partitionDriveNumber>
      <driveType>[value]</driveType>
      <connection>
         <connectionStatus>[value]</connectionStatus>
         <hostID>[value]</hostID>
         <portID>[value]</portID>
      </connection>
      <serialNumber>[value]</serialNumber>
      <manufacturerSerialNumber>[value]</manufacturerSerialNumber>
      <driveFirmware>[value]</driveFirmware>
      <dcmFirmware>[value]</dcmFirmware>
      <wwn>[value]</wwn>
      <fibreAddress>soft|hard</fibreAddress>
      <loopNumber>[value]<loopNumber>
      <sparedWith>[value]</sparedWith>
      <spareFor>[value]</spareFor>
      <sparePotential>[value]</sparePotential>
      <health>[value]</health>
      <firmwareStaging>
         <firmware>[value]</firmware>
         <complete>yes|no</complete>
         <percentStaged>[value]</percentStaged>
         <committing>yes|no</committing>
      </firmwareStaging>
   </drive>
   ...
</driveList>
```

where the value for:

This parameter...	Indicates...
ID	The component identifier (ID) assigned to the drive by the library. The component identifier uses the form **FR[*integer*]/DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the drive is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the drive. Not used in T120 library component identifiers. ▪ **[*interface*]** = The interface used by the drive. Values: ▪ **f** = Fibre Channel ▪ **s** = Serial Attached SCSI (SAS) ▪ (blank) = SCSI ▪ **[*technology*]** = The technology used by the drive. Values: **LTO**, **TS11x0, T10K** ▪ **DRV[*integer*]** = The designator of the drive bay in the DBA, as viewed from the back of the library. For all libraries except the T120, the value of ***x*** can be **1** through **4**. For the T120 library, the value of ***x*** can be **1** through **6** for full-height drives, and **1b** through **6a** for half-height drives. **Notes:** ▪ The **ID** values returned by the **driveList.xml** command without any parameters are the component identifiers for the drives currently installed in the library (see driveList.xml on page 33). ▪ Only the T120 library supports SAS drives. ▪ Only the T380, T950, and TFinity libraries support **TS11x0** technology drives. ▪ Only TFinity libraries support **T10K** drives. **EXAMPLE:** In the example command response on page 38, the **ID** for the first drive is **FR1/DBA1/fLTO-DRV1**, indicating that it is a Fibre Channel LTO drive installed in drive bay 1 of DBA1 in Frame 1. See "Drive Identifiers" in your library *User Guide* for additional information about drive component identifiers.
driveStatus	The status of the drive as reported by the library. Values: **OK**, **impaired**, **missing**, or **unknown**.
partition	The name of the partition to which the drive is currently assigned. **Note:** This parameter is not included if the drive is not assigned to a partition.
partitionDrive Number	The logical number of the drive within the partition. **Note:** This parameter is not included if the drive is not assigned to a partition.
driveType	The drive technology, generation, and connection type for the drive. Values: ▪ **IBM Ultrium-TD*n***, where *n* is the generation = LTO drives ▪ **IBM 3592E*nn* Fibre**, where ***nn***=07 indicates TS1140 technology, ***nn***=08 indicates TS1150 technology, ***nn***=55 indicates TS1155 technology, and ***nn***=60 indicates TS1160 technology (T380, T950, and TFinity libraries) ▪ **Oracle/STK T10K** = T10K drives (TFinity libraries)

This parameter...	Indicates...
connection (T120 only)	The connection type. Values: ▪ **hostID** = The hostID of the host to which the direct-attached drive is connected. ▪ **portID** = The portID of the F-QIP port through which the drive is connected to the host. ▪ **connectionStatus** = The status of the drive connection. Values: ▪ **Connected to Host** = The drive is configured to be connected to the host using a direct-attached connection. ▪ **Connected to Port** = The drive is configured to be connected to the host through an F-QIP port. ▪ **Not Connected** = The drive is not configured to be connected to the host. ▪ **Unknown** = The status of the drive connection is unknown. **Note:** This parameter is not returned for T10K drives.
serialNumber	The location-based serial number assigned to the drive while it is in the library. This is the serial number reported to the host for the drive. Using a location-based serial number makes it possible to replace one drive with another without having to reconfigure the storage management software that accesses the drive. **Note:** This parameter is not returned for T10K drives.
manufacturerSerialNumber	The serial number assigned to the physical drive by the drive manufacturer. This serial number is shown in MLM and DLM reports and is also used for tracking the drives when they are not inside the library. **Note:** This parameter is set to **Unavailable** if the library is unable to retrieve the manufacturer's serial number from the drive.
driveFirmware	The firmware version in use by the drive. Values: ▪ ***version*** = The firmware version currently in use by the drive. ▪ **Unknown** = The library is not able to determine the firmware version. ▪ **(powered off)** = The drive is powered off and cannot return its firmware information.
dcmFirmware	The firmware version being used by the drive sled that houses the drive.
wwn	The World Wide Name (WWN) for the drive. See "Drive Connectivity" in your library *User Guide* for detailed information about the WWNs for drives. **Notes:** ▪ This data is only returned for Fibre Channel or Serial Attached SCSI (SAS) drives. SAS drives are only supported in the T120 library. ▪ The WWN is actually the WWPN for port A on the drive sled. The WWPN for port B is the same as the one for port A except that the second digit from the left is **2** instead of **1**. ▪ This parameter is not returned for T10K drives.

This parameter...	Indicates...
fibreAddress	Whether the drive is configured for soft addressing or an assigned Loop ID. Values: ▪ **soft** = The drive uses soft addressing. ▪ **hard** = The drive uses an assigned Loop ID. **Notes:** ▪ This parameter is not included if the drive is not assigned to a partition. ▪ This parameter is not returned for T10K drives.
loopNumber	The assigned loop ID. **Note:** This parameter is only included if the drive is assigned to a partition and **fibreAddress** is **hard.**
sparedWith	The library-assigned component identifier of the Global Spare drive used to replace the drive. See "Drive Identifiers" and "Using a Global Spare Drive" in your library *User Guide* for more information. **Note:** This parameter is only returned if the drive has been replaced by a Global Spare drive.
spareFor	The library-assigned component identifier of the drive that the Global Spare drive is replacing. See "Drive Identifiers" and "Using a Global Spare" in your library *User Guide* for more information. **Note:** This parameter is only returned if the drive is a Global Spare being used to replace another drive.
sparePotential	Whether the drive is configured for use as a Global Spare drive for a partition and is available for use. See "Assign Global Spare Drives" in your library *User Guide* for information about configuring a drive as a Global Spare for a partition. Value: **defined** = The drive is configured as a Global Spare but is not currently in use as a spare. **Note:** This parameter is only returned if the drive is configured as a Global Spare but is not currently being used to replace another drive.
health	The drive health. The health can be one of the following: **Green**, **Yellow**, **Red**, or **Unknown**. See "Monitoring Drive Health Using DLM" in your library *User Guide* for detailed information about Drive Lifecycle Management (DLM) and drive health. **Notes:** ▪ This parameter is only returned if the drive is assigned to a partition. ▪ This parameter is not returned for T10K drives.
firmwareStaging	The status of the firmware staging process. **Note:** Firmware staging is only available for LTO-5 and later generation and TS1140 technology and later generation drives. Values: ▪ **firmware** = The drive firmware being staged or committed. ▪ **complete** = The status of the staging process. Values: **Yes** = Staging is complete; **No** = Staging is in process. ▪ **percentStaged** = The percentage of the firmware already staged. ▪ **committing** = The status of updating the drive using the staged firmware. Values: **Yes** = Committing is in process; **No** = Committing has not started.

Example Command and Response The following command:

```
driveList.xml
```

retrieves the following information for a T950 library that has two Fibre Channel LTO-3 drives installed, both of which are in a partition named LTO Partition.

```
<driveList>
   <drive>
      <ID>FR1/DBA1/fLTO-DRV1</ID>
      <driveStatus>OK</driveStatus>
      <partition>LTO Partition</partition>
      <partitionDriveNumber>1</partitionDriveNumber>
      <driveType>IBM Ultrium-TD3 Fibre</driveType>
      <serialNumber>1011000EC2</serialNumber>
      <manufacturerSerialNumber>
         10380861
      </manufacturerSerialNumber>
      <driveFirmware>93G0</driveFirmware>
      <dcmFirmware>4.7.0</dcmFirmware>
      <wwn>21 11 00 90 A5 00 0E C2</wwn>
      <fibreAddress>soft</fibreAddress>
      <health>Green</health>
   </drive>
   <drive>
      <ID>FR1/DBA1/fLTO-DRV2</ID>
      <driveStatus>OK</driveStatus>
      <partition>LTO Partition</partition>
      <partitionDriveNumber>2</partitionDriveNumber>
      <driveType>IBM Ultrium-TD3 Fibre</driveType>
      <librarySerialNumber>1012000EC2</librarySerialNumber>
      <manufacturerSerialNumber>
         10241727
      </manufacturerSerialNumber>
      <driveFirmware>93G0</driveFirmware>
      <dcmFirmware>4.7.0</dcmFirmware>
      <wwn>21 12 00 90 A5 00 0E C2</wwn>
      <fibreAddress>soft</fibreAddress>
      <health>Green</health>
   </drive>
</driveList>
```

generateDriveTraces

Description Generate a new drive trace file.

Notes:
- This action was added with BlueScale12.4.1.
- This action is not supported for TS11*xx* technology drives.
- This action is not supported for T10K drives.
- This action can only be used for LTO-5 and later generation drives.

Syntax

```
driveList.xml?action=generateDriveTraces&
driveTracesDrives=[AllDrives|[Drive ID],[Drive ID],...]
[&extraInformation=[string]]
```

where the value for:

This parameter...	Specifies...

driveTracesDrives	The drive(s) you want to generate a drive trace.
	Values: **AllDrives, [*Drive ID x*]** where:
	▪ **AllDrives** = Generate traces for all LTO-5 and later generation drives.
	▪ ***Drive ID*** = Is the component identifier for the drive you want to generate a drive trace, using the form **FR[*integer*]/DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*]**, where:
	  ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the drive is in a library that supports multiple frames.
	  ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the drive. Not used in T120 library component identifiers.
	  ▪ **[*interface*]** = The interface used by the drive.
	  Values:
	  ▪ **f** = Fibre Channel
	  ▪ **s** = Serial Attached SCSI (SAS)
	  ▪ (blank) = SCSI
	  ▪ **[*technology*]** = The technology used by the drive.
	  Values: **LTO**
	  ▪ **DRV[*integer*]** = The designator of the drive bay in the DBA, as viewed from the back of the library. For all libraries except the T120, the value of ***x*** can be **1** through **4**. For the T120 library, the value of ***x*** can be **1** through **6** for full-height drives, and **1b** through **6a** for half-height drives.
	Notes:
	▪ The ID values returned by the **driveList.xml** command without any parameters are the component identifiers for the drives currently installed in the library (see driveList.xml on page 33).
	▪ See "Drive Identifiers" in your library *User Guide* for additional information about drive component identifiers.

extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<driveList>
   <status>OK</status>
   <message>[message text}</message>
</driveList>
```

Progress Use `driveList.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
driveList.xml?action=generateDriveTraces&driveTracesDrives=FR1/
DBA1/fLTO-DRV1
```

immediately returns the following XML-formatted data:

```
<driveList>
   <status>OK</status>
   <message>Started drive traces creation. Set progress in your
      query for status.</message>
</driveList>
```

and generates a drive trace for drive 1 in DBA1. When the command is complete, the response to `driveList.xml?progress` contains `<status>OK</status>`.

getDriveLoadCount

Description Returns the number of times that a tape was loaded into the specified drive.

Notes:
- This action was added with BlueScale12.7.00.
- This action is not supported for T10K drives.

Syntax `driveList.xml?action=getDriveLoadCount&driveName=<drive>`

where the value for:

This parameter...	Specifies...
driveName	The component identifier (ID) assigned to the drive by the library. The component identifier uses the form **FR[*integer*]/DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the drive is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the drive. Not used in T120 library component identifiers. ▪ **[*interface*]** = The interface used by the drive. Values: ▪ **f** = Fibre Channel ▪ **s** = Serial Attached SCSI (SAS) ▪ (blank) = SCSI ▪ **[*technology*]** = The technology used by the drive. Values: **LTO**, **TS11x0** ▪ **DRV[*integer*]** = The designator of the drive bay in the DBA, as viewed from the back of the library. For all libraries except the T120, the value of ***x*** can be **1** through **4**. For the T120 library, the value of ***x*** can be **1** through **6** for full-height drives, and **1b** through **6a** for half-height drives. **Notes:** ▪ The **ID** values returned by the **driveList.xml** command without any parameters are the component identifiers for the drives currently installed in the library (see [no parameters] or list on page 34). ▪ See "Drive Identifiers" in your library *User Guide* for additional information about drive component identifiers. ▪ Only the T120 library supports SAS drives. ▪ Only the T380, T950, and TFinity libraries support **TS11x0** technology drives.

Command Response The command returns the following XML-formatted data:

```
<getDriveLoadCount>
   <loadCount>[value]</loadCount>
</getDriveLoadCount>
```

where the value for:

This parameter...	Indicates...
loadCount	The number of times a tape was loaded into the drive in its lifetime.

Example Command The following command

```
driveList.xml?action=getDriveLoadCount&driveName=FR3/DBA4/fTS11x0-DRV3
```

retrieves the drive load count for the TS11x0 technology drive in frame 3, DBA4, drive location 3.

```
<getDriveLoadCount>
   <loadCount>1283</loadCount>
</getDriveLoadCount>
```

getDriveTraces

Description Retrieves the last drive trace file generated by the generateDriveTraces action (see generateDriveTraces on page 39).

Note: This action was added with BlueScale12.4.1.

Syntax

```
driveList.xml?action=getDriveTraces&
driveTracesGetType=[email|download|saveToUSB]&
[emailAddress=[Mail Recipient]]
```

where the value for:

This parameter...	Specifies...
driveTracesGetType	Where you want to save the drive trace file. Values: ▪ **email** = Sends the file to the mail recipient specified by **emailAddress**. ▪ **download** = Downloads the file to the computer you are using to access the library. ▪ **saveToUSB** = Saves the file to a USB device that is connected to the LCM. **Note:** If you want to save the drive trace file to a USB device, make sure that the USB device is connected to the LCM before running the command.
emailAddress (optional)	The email address of an already-configured mail recipient to whom the library emails the drive trace file. **Notes:** ▪ Do not send the drive trace to *autosupport@spectralogic.com*. Spectra Logic does not save emailed drive trace files unless they are specifically requested for troubleshooting. ▪ See "Configure Mail Users" in you library *User Guide* for information about configuring mail recipients. ▪ Only provide **emailAddress** if **driveTracesGetType** is **email**.

Command Response If you specify **driveTracesGetType=download**, the command immediately returns a ZIP file containing the drive trace. Otherwise, the command returns the following XML-formatted data:

```
<driveList>
   <status>OK</status>
   <message>[success message text]</message>
</driveList>
```

Example Command The following command emails the drive trace ZIP file to the already configured mail recipient YourName@company.com.

```
driveList.xml?action=getDriveTraces&driveTracesGetType=email&
emailAddress=YourName@company.com
```

prepareToReplaceDrive

Description Prepares the specified drive for replacement by taking it offline. The drive sled LED flashes orange and DLM marks the drive as impaired. After the command completes successfully, the drive can be safely removed from the library and a replacement drive installed.

Syntax

```
driveList.xml?action=prepareToReplaceDrive&
   driveName=[drive to replace]
```

where the value for:

This parameter...	Specifies...
driveName	The component identifier (ID) assigned to the drive by the library. The component identifier uses the form **FR[*integer*]/DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the drive is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the drive. Not used in T120 library component identifiers. ▪ **[*interface*]** = The interface used by the drive. Values: ▪ **f** = Fibre Channel ▪ **s** = Serial Attached SCSI (SAS) ▪ (blank) = SCSI ▪ **[*technology*]** = The technology used by the drive. Values: **LTO**, **TS11x0, T10K** ▪ **DRV[*integer*]** = The designator of the drive bay in the DBA, as viewed from the back of the library. For all libraries except the T120, the value of ***x*** can be **1** through **4**. For the T120 library, the value of ***x*** can be **1** through **6** for full-height drives, and **1b** through **6a** for half-height drives. **Notes:** ▪ The **ID** values returned by the **driveList.xml** command without any parameters are the component identifiers for the drives currently installed in the library (see [no parameters] or list on page 34). ▪ See "Drive Identifiers" in your library *User Guide* for additional information about drive component identifiers. ▪ Only the T120 library supports SAS drives. ▪ Only the T380, T950, and TFinity libraries support **TS11x0** technology drives. ▪ Only TFinity libraries support **T10K** drives.

Command Response The command returns the following XML-formatted data:

```
<prepareToReplaceDrive>
   <status>OK</status>
   <message>
      prepareToReplaceDrive completed for drive driveName
   </message>
</prepareToReplaceDrive>
```

Example Command and Response The following command:

```
driveList.xml?action=prepareToReplaceDrive&
   driveName=FR2/DBA1/fLTO-DRV2
```

immediately returns the following XML-formatted data:

```
<prepareToReplaceDrive>
   <status>OK</status>
   <message>
      prepareToReplaceDrive completed for drive FR2/DBA1/fLTO-DRV2
   </message>
</prepareToReplaceDrive>
```

and prepares the Fibre Channel LTO drive in drive bay 2 of DBA1 located in frame 2 of a T950 library to be replaced.

resetDrive

Description Resets the specified drive by power cycling it.

Syntax `driveList.xml?action=resetDrive&driveName=[drive to reset][&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
driveName	The component identifier (ID) assigned to the drive by the library. The component identifier uses the form **FR[*integer*]/DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the drive is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the drive. Not used in T120 library component identifiers. ▪ **[*interface*]** = The interface used by the drive. Values: ▪ **f** = Fibre Channel ▪ **s** = Serial Attached SCSI (SAS) ▪ (blank) = SCSI ▪ **[*technology*]** = The technology used by the drive. Values: **LTO**, **TS11x0**, **T10K** ▪ **DRV[*integer*]** = The designator of the drive bay in the DBA, as viewed from the back of the library. For all libraries except the T120, the value of ***x*** can be **1** through **4**. For the T120 library, the value of ***x*** can be **1** through **6** for full-height drives, and **1b** through **6a** for half-height drives. **Notes:** ▪ The **ID** values returned by the **driveList.xml** command without any parameters are the component identifiers for the drives currently installed in the library (see [no parameters] or list on page 34). ▪ See "Drive Identifiers" in your library *User Guide* for additional information about drive component identifiers. ▪ Only the T120 library supports SAS drives. ▪ Only the T380, T950, and TFinity libraries support **TS11x0** technology drives. ▪ Only TFinity libraries support **T10K** drives.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<resetDrive>
   <status>OK</status>
   <message>[message text]</message>
</resetDrive>
```

Progress Use `driveList.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), the drive is then ready to resume operation. If desired, use the **systemMessages.xml** command (see systemMessages.xml on page 186) to retrieve any system messages generated as a result of the reset.

Example Command and Response The following command:

```
driveList.xml?action=resetDrive&driveName=FR1/DBA1/fLTO-DRV1
```

immediately returns the following XML-formatted data:

```
<resetDrive>
   <status>OK</status>
   <message>resetDrive started for drive FR1/DBA1/fLTO-DRV1. Set
      progress in your query for status.</message>
</resetDrive>
```

and resets drive 1 in DBA1. When the command is complete, the response to `driveList.xml?progress` contains `<status>OK</status>`.

CHAPTER 5

encryption

encryption.xml

Use the **encryption.xml** command to log into the library's encryption feature.

login

Description Logs into the encryption feature using the specified encryption user password. Refer to the *Spectra Encryption User Guide* for detailed information about configuring and using either Spectra SKLM, KMIP, or BlueScale Encryption key management.

Important You must first log into the library as a user with superuser privileges using the **login.xml** command (see login.xml on page 98) before you can log into the BlueScale Encryption application.

Note: Before you can use encryption with a storage partition, encryption must be enabled for the partition using the BlueScale user interface.

You remain logged into the encryption feature until you terminate the current connection to the library or log in again.

Syntax `encryption.xml?action=login&encryptionPassword=[password]`

where the value for:

This parameter...	Specifies...
encryptionPassword	The encryption user password. The value for this parameter is blank if no password has been set.

Command Response The command immediately returns the following XML-formatted data:

```
<encryption>
   <status>OK</status>
</encryption>
```

Example Command The following command:

```
encryption.xml?action=login&encryptionPassword=encrypt1
```

logs into the encryption feature using the encryption password `encrypt1`.

CHAPTER 6

etherLibStatus

etherLibStatus.xml

Use the **etherLibStatus.xml** command to check and then retrieve the status of the library's EtherLib connections.

Notes:
- EtherLib is only supported on T950 and TFinity libraries.
- This command was added with BlueScale12.6.45.

Action	
list	below
refresh	page 49

list

Description Retrieves the EtherLib status information gathered with the **refresh** action (see refresh on page 49).

Syntax `etherLibStatus.xml?action=list`

Command Response The command immediately returns the following XML-formatted data:

```
<etherLibStatus>
   <component>
      <ID>LCM</ID>
      <connection>
         <target>[Id]</target>
         <connected>yes|no</connected>
      </connection>
      ...
   </component>
</etherLibStatus>
```

where the value for:

This parameter...	Indicates...
component	The container for information about one initiating component.
ID	The identifier of the initiating component. Value: **LCM**
target	The identifier of the target component. For RCMs the format of the identifier is **FR[*integer*]/RCM**, where **FR[*integer*]** is the identifier of the frame in which the RCM is located.
connected	Whether the target is (**yes**) or is not (**no**) connected to the initiating component. Values: **yes**, **no**

Example Command and Response The following command:

```
etherLibStatus.xml?action=list
```

immediately returns the following XML-formatted data when issued to a single frame T950 library with EtherLib installed:

```
<etherLibStatus>
   <component>
      <ID>LCM</ID>
      <connection>
         <target>FR1/RCM</target>
         <connected>yes</connected>
      </connection>
   </component>
</etherLibStatus>
```

refresh

Description Attempts to reestablish the Ethernet connection and update the stored status information for each EtherLib connection.

Syntax

```
etherLibStatus.xml?action=refresh[&extraInformation=[string]]
```

where the value for:

This parameter...	Specifies...
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command immediately returns the following XML-formatted data:

```
<etherLibStatus>
   <message>[message text]</message>
   <status>OK</status>
</etherLibStatus>
```

Progress Use `etherLibStatus.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
etherLibStatus.xml?action=refresh
```

immediately returns the following XML-formatted data:

```
<etherLibStatus>
   <message>Started Etherlib status refresh action. Set progress
      in your query for status.</message>
   <status>OK</status>
</etherLibStatus>
```

When the status refresh is complete, the response to `etherLibStatus.xml?progress` contains `<status>OK</status>`.

CHAPTER 7

HHMData

HHMData.xml

Use the **HHMData.xml** command to retrieve the current status of the Hardware Health Monitoring (HHM) counters for the library and to reset certain counters to zero after performing the necessary hardware maintenance procedures. You can also set the threshold level at which certain counters send a reminder that maintenance is needed. See "View Hardware Health Monitoring (HHM) Data" in your library *User Guide* for additional information about HHM.

Note: The parameters used in the **HHMData.xml** commands depend on the library type. To determine the parameters for your library, use the **HHMData.xml** command without any parameters.

Action	
[no parameters] or list	page 51
resetCounterData	page 54
setThresholdData	page 56

[no parameters] or list

Description Returns a report showing the current data for all of the HHM counters for the library.

Syntax `HHMData.xml?action=list`

Command Response The command immediately returns the following XML-formatted data:

Note: The names and number of the HHM counters, as well as the reminders for the Trip1 and Trip2 sub-counters, differ depending on the types of HHM counters supported by the library.

```
<HHMData>
   <counter>
      <typeName>
         Horizontal Axis|Vertical Axis|Picker Axis|Toggle Axis|
         Rotational Axis|Side Axis|General Maintenance|
         Drive to Drive Move|Drive to Slot Move|Slot to Slot Move|
         Slot to Drive Move|TAP In Move|TAP Out Move
      </typeName>
      <subType>
         <typeName>Life|Trip1|Trip2</typeName>
         <value>[value]</value>
         <unit>[unit of measure]</unit>
         <reminder>
            <typeName>[reminder name]</typeName>
            <severity>low|medium|high</severity>
            <defaultThreshold>[value]</defaultThreshold>
            <currentThreshold>[value]</currentThreshold>
            <postedDate>[date]</postedDate>
         </reminder>
      </subtype>
      ...
   </counter>
   ...
</HHMData>
```

where the value for:

This parameter...	Indicates...
typeName (HHM counter)	The name of the HHM counter for which data is being returned. Values: **Horizontal Axis**, **Vertical Axis**, **Picker Axis**, **Toggle Axis**, **Rotational Axis**, **Side Axis**, **General Maintenance**, **Drive to Drive Move**, **Drive to Slot Move, Slot to Slot Move**, **Slot to Drive Move**, **TAP In Move**, **TAP Out Move** **Notes:** ▪ The XML data returned in the command response contains sections for each HHM counter and its associated subType counters and reminders. ▪ The number of the HHM counters and their names depend on the library type.

This parameter...	Indicates...
typename (subType)	The name of the subType counter for which data is being returned. All of the HHM counters have one or more associated subType counters. Values: ▪ **Life** = Tracks the counter data over the lifetime of the library. ▪ **Trip1** and **Trip2** = Track specific aspects of the HHM counter. These counters may or may not be present. ▪ **None** = The HHM counter does not have any of the other subType counters associated with it. **Note:** For the TFinity library, many of the HHM counters contain two sets of **Life**, **Trip1**, and **Trip2** subType counters, one set for **Robot 1** and the other for **Robot2** (TeraPorter 1 and TeraPorter 2, respectively).
value	The current value of the counter.
unit	The unit of measure for that data. Values= **inches**, **degrees**, **steps**, **hours**, **moves.**
typeName (reminder)	The reminders (system messages) that have been posted for the **Trip1** and **Trip2** subType counters associated with the current HHM counter. When one of these counters reaches the threshold value set for the counter, the library adds a reminder to the HHM data returned by the **HHMData.xml** command and turns on the HHM icon in the status bar on the BlueScale user interface. Each reminder includes the following parameters: ▪ **typeName** = The name of the reminder. Values = **Service HAX**, **Check Contact Brushes** (TFinity library only), **Service HAX Belt** (all except T120 and TFinity libraries), **Service VAX**, **Service VAX Belt**, **Service VAX Cable** (T120 only), **Service Transporter** (all except T120), **Service Required** **Notes:** ▪ The **Life** counter does not have a reminder threshold value associated with it and cannot be reset to zero. For this reason, the command response data for the **Life** counter does not include a **reminder** section. ▪ Not all **Trip1** and **Trip2** subType counters have an associated reminder threshold value. ▪ If the threshold for the subType counter has not been reached, the reminder section is not present. ▪ The reminder threshold values for the **Trip1** and **Trip2** subType counters are set using the **setThresholdData** command (see setThresholdData on page 56). ▪ After the maintenance action indicated in the HHM reminder has been completed, the current value of the reminder can be reset to zero (see resetCounterData on page 54).
severity	The urgency of the reminder. Values = **low**, **medium**, **high**. See "Check and Respond to Messages" in your library *User Guide* for a description of these system messages.
defaultThreshold	The value of the factory default setting for the reminder threshold.
currentThreshold	The value of the current reminder threshold (see setThresholdData on page 56 for information about setting the reminder threshold).
postedDate	The date on which the reminder for a counter was posted. If a reminder has not been posted, the date is set to **None**.

Example Command and Response The following command:

```
HHMData.xml
```

retrieves the following HHM data:

Note: The following example shows part of the HHM data returned by a TFinity library. The actual data returned depends on the library type.

```
<HHMData>
   <counter>
      <typeName>Horizontal Axis</typeName>
      <subType>
         <typeName>Life (Robot 1)</typeName>
         <value>25172883</value>
         <unit>inches</unit>
      </subType>
      <subType>
         <typeName>Trip1 (Robot 1)</typeName>
         <value>49513396</value>
         <unit>inches</unit>
         <reminder>
            <typeName>Service HAX</typeName>
            <severity>low</severity>
            <defaultThreshold>35950000</defaultThreshold>
            <currentThreshold>35950000</currentThreshold>
            <postedDate>2011/08/09 17:08:04</postedDate>
         </reminder>
      </subType>
      <subType>
         <typeName>Trip2 (Robot 1)</typeName>
         <value>9295692</value>
         <unit>inches</unit>
         <reminder>
            <typeName>Check Contact Brushes</typeName>
            <severity>low</severity>
            <defaultThreshold>46300000</defaultThreshold>
            <currentThreshold>46300000</currentThreshold>
            <postedDate>None</postedDate>
         </reminder>
      </subType>
      <subType>
         <typeName>Life (Robot 2)</typeName>
         <value>25172883</value>
         <unit>inches</unit>
      </subType>
      ...
   </counter>
   ...
</HHMData>
```

resetCounterData

Description Resets the specified HHM counter to zero. A counter is typically reset to zero following the completion of the regularly scheduled standard maintenance of the component.

Caution

Do not run this command unless you are specifically directed to do so by Spectra Logic Support. Changing the counter values can result in components not receiving regularly scheduled maintenance when it is due.

Notes:

- Only one subType counter of one HMM counter can be reset in each **resetCounterData** command. To reset multiple counters, you must issue separate commands.
- The following syntax is for a TFinity library. The syntax for other libraries does not include the **robot** parameter.
- This command corresponds to the HHM: Set Counters advanced utility in the BlueScale user interface.

Syntax

```
HHMData.xml?action=resetCounterData&type=[Horizontal
Axis|Vertical Axis|Picker Axis|Toggle Axis|Rotational Axis|Side
Axis|Drive to Drive Move|Drive to Slot Move|Slot to Slot Move|Slot
to Drive Move|TAP In Move|TAP Out Move]&
subType=[Trip1|Trip2|None]&robot=[Robot 1|Robot 2]
```

where the value for:

This parameter...	Specifies...
type	The name of the HHM counter for which the subType counter specified by the command is to be reset. Values: **Horizontal Axis, Vertical Axis, Picker Axis, Rotational Axis, Magazine Axis, Toggle Axis, Side Axis, Drive to Drive Move, Drive to Slot Move, Slot to Slot Move, Slot to Drive Move, TAP In Move, TAP Out Move**.
subType	The name of the subType counter to be reset. Values: **Trip1**, **Trip2**, **None** **Notes:** ▪ The **Life** subType counter cannot be reset. ▪ Most of the counters have associated **Trip1** or **Trip2** subType counters that can be reset. ▪ For the TFinity library, many of the HHM counters contain two sets of **Trip1** and **Trip2** subType counters, one set for **Robot 1** and the other for **Robot2** (TeraPorter 1 and TeraPorter 2, respectively). Each of the counters must be reset independently.
robot	**TFinity library only.** For which robot the **Trip1** or **Trip2** counter is being reset. Values: **Robot 1** or **Robot 2**.

Command Response The command immediately returns the following XML-formatted data:

```
<HHMData>
   <resetCounterData>
      <status>[OK|FAILURE]</status>
      <message>["Successfully set|"Unable to set]
         '[type] [subType]' to '0'."</message>
   </resetCounterData>
</HHMData>
```

Example Command and Response

The following command resets the **Trip1** subType counter for the **Horizontal Axis** counter:

```
HHMData.xml?action=resetCounterData&type=Horizontal Axis&
   subType=Trip1
```

and returns the following:

```
<HHMData>
   <resetCounterData>
      <status>OK</status>
      <message>
         Successfully set 'Horizontal Axis Trip1 counter' to '0'.
      </message>
   </resetCounterData>
</HHMData>
```

setThresholdData

Description Changes the reminder threshold for those HHM counters that have a **Trip 1** or **Trip 2** subType counter associated with them.

When the **Trip1** or **Trip2** counter reaches the specified threshold, the library adds a reminder to the HHM data returned by the **HHMData.xml** command (see typeName on page 52) and turns on the HHM icon in the status bar on the BlueScale user interface.

Caution

Do not run this command unless you are specifically directed to do so by Spectra Logic Support. Changing the threshold values can result in components not receiving regularly scheduled maintenance at the appropriate intervals.

Syntax `HHMData.xml?action=setThresholdData&event=[Service HAX|Check Contact Brushes|Service HAX Belt|Service VAX|Service VAX Belt|Service VAX Cable|Service Transporter|Service Required]&default=[true|false]&value=[value]`

Notes:
- Only HHM counters that have a **Trip 1** or **Trip 2** subType counter have configurable reminder thresholds.
- Only one reminder threshold value can be set with each command. To reset multiple thresholds, you must issue separate commands.
- For the TFinity library, the reminder threshold for the **Trip 1** or **Trip 2** subType counter is set to the same value for both robots.

where the value for:

This parameter...	Specifies...
event	The name of the reminder for which the threshold is being set. Values: **Service HAX**, **Check Contact Brushes** (TFinity library only), **Service HAX Belt** (all except T120 and TFinity libraries), **Service VAX**, **Service VAX Belt**, **Service VAX Cable** (T120 only), **Service Transporter** (all except T120), **Service Required**
default	Whether the reminder threshold uses the factory default value or the threshold set by the value parameter. Values: **true** (use the factory default threshold), **false** (use the manually set threshold) **Note:** If **default** is set to **false**, then the **value** parameter must contain a valid number. In this case, it is not necessary to also use the **default** parameter in the command.

This parameter...	Specifies...
value	The value for the reminder threshold if the value for the **default** parameter is set to **false**. Values: **1** through **4294967295** **Notes:** ▪ If the **default** parameter is set to **true**, then it is not necessary to include the **value** parameter in the command. ▪ The unit of measure for the events depends on the type of counter. For example, for the **Service HAX** counter, the unit of measure for the reminder threshold is inches traveled; for the **Service Required** counter, the unit of measure for the reminder threshold is elapsed minutes of operation. See the **unit** parameter on page 52 for additional information.

Command Response The command immediately returns the following XML-formatted data:

```
<HHMData>
   <setThresholdData>
      <status>[OK|FAILURE]</status>
      <message>["Successfully set|"Unable to set] '[event]'
         threshold to '[default|value]'."
      </message>
   </setThresholdData>
</HHMData>
```

Example Command and Response The following command:

```
HHMData.xml?action=setThresholdData&event=Service HAX Belt&
   default=false&value=46300000
```

returns the following:

```
<HHMData>
   <setThresholdData>
      <status>OK</status>
      <message>
         Successfully set 'Service HAX Belt' to '46300000'.
      </message>
   </setThresholdData>
</HHMData>
```

CHAPTER 8

inventory

inventory.xml

Use the **inventory.xml** command to retrieve information about the inventory status of all the slots and drives assigned to a specified partition or to audit the inventory of a specific TeraPack magazine.

audit

Description A TeraPack audit compares the database inventory of a TeraPack magazine to the inventory discovered by a barcode scan of the magazine. In the event of a mismatch, the inventory database is updated with the results of the scan.

Notes:
- This action is only supported on TFinity libraries.
- This action was added with BlueScale12.7.00.01.

Syntax `inventory.xml?action=audit&partition=<partition name>&elementType=<storage|IE>&TeraPackOffset=<offset>[&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
partition	The exact name of the partition containing the TeraPack magazine to audit. **Notes:** ▪ The partition name is case-sensitive. ▪ Use the **partitionList.xml** command to retrieve a list of all the partitions currently configured in the library (see partitionList.xml on page 171). ▪ The partition name is set when the partition is created. Refer to your library *User Guide* for detailed information about configuring and using partitions in the library.

This parameter...	Specifies...
elementType	The type of element in which the magazine is contained. Values: ▪ **storage** - The storage pool for the partition. ▪ **IE** - The Entry/Exit pool of the partition.
TeraPackOffset	The offset value for the magazine to inventory. The offset identifies the virtual location of the TeraPack magazine in the partition inventory. Values: Use `physInventory.xml?action=partition` to obtain the offset value for the inventory locations you want to use (see physInventory.xml on page 172). **Important:** The **offset** values given by **physInventory.xml** are one-based. The **TeraPackOffset** required by **inventory.xml?action=audit** is zero-based. You must subtract 1 from the **offset** value before supplying it as a **TeraPackOffset**.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<inventory>
   <status>OK</status>
   <message>[message text]</message>
</inventory>
```

Progress Use `inventory.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
inventory.xml?action=audit&partition=partition1
&elementType=storage&TeraPackOffset=1
```

immediately returns the following XML-formatted data:

```
<inventory>
   <status>OK</status>
   <message>
      Started TeraPack audit. Set progress in your query for
      status.
   </message>
</inventory>
```

When the response to `inventory.xml?progress` contains `<status>OK</status>`, the command is complete. Use the `inventory.xml?action=getAuditResults` command (see getAudit Results) to retrieve the audit results.

getAudit Results

Description Retrieves the audit results collected by the command `inventory.xml?action=audit` command (see audit on page 58). The audit results can only be retrieved once.

Notes:
- This action is only supported on TFinity libraries.
- This action was added with BlueScale12.7.00.01.

Syntax `inventory.xml?action=getAuditResults`

Command Response The command immediately returns the following XML-formatted data:

```
<inventory>
   <auditResults>
      <elementType>storage|IE</elementType>
      <offset>[value]</offset>
      <barcode>[value]</barcode>
      <contentsMatch>yes|no</contentsMatch>
      <expectedContents>
         <slot>
            <number>[value]</number>
            <barcode>[value]</barcode>
         </slot>
         ...
      </expectedContents>
      <actualContents>
         <slot>
            <number>[value]</number>
            <barcode>[value]</barcode>
         </slot>
         ...
      </actualContents>
   </auditResults>
</inventory>
```

where the value for:

This parameter...	Indicates...
elementType	The type of element in which the magazine is contained. Values: ▪ **storage** - The storage pool for the partition. ▪ **IE** - The Entry/Exit pool of the partition.
offset	The offset value for the magazine audited. The offset identifies the virtual location of the TeraPack magazine in the partition inventory.
barcode	The barcode on the TeraPack magazine.
contentsMatch	Whether the inventory discovered during the scan matches the inventory in the data base. Values: **yes**, **no**
expectedContents	The container for the inventory according to the database. **Note:** This section is only returned if **contentsMatch** is **no**.
actualContents	The container for the inventory according to the TeraPack magazine scan.
slot	The container for information about one tape in the magazine. **Note:** Only full slots are listed.

This parameter...	Indicates...
number	The slot in the magazine containing the tape. Slot numbers are one-based.
barcode	The barcode on the tape.

Example Command The following command:

```
inventory.xml?action=getAuditResults
```

retrieves the following audit results:

```
<inventory>
   <auditResults>
      <elementType>storage</elementType>
      <offset>1</offset>
      <barcode>LU83567</barcode>
      <contentsMatch>no</contentsMatch>
      <expectedContents>
         <slot>
            <number>1</number>
            <barcode>000380L5</barcode>
         </slot>
         <slot>
            <number>3</number>
            <barcode>000382L5</barcode>
         </slot>
      </expectedContents>
      <actualContents>
         <slot>
            <number>2</number>
            <barcode>000380L5</barcode>
         </slot>
         <slot>
            <number>3</number>
            <barcode>000382L5</barcode>
         </slot>
      </actualContents>
   </auditResults>
</inventory>
```

getMoveResult

Description Provides the result of an `inventory.xml?action=move`.

Note: This action was added with BlueScale12.8.04.

Syntax `inventory.xml?action=getMoveResult&partition=[`*partition name*`]`

Command Response The command immediately returns the following XML-formatted data:

```
<inventory>
   <moveResult>
      <message>
   </moveResult>
</inventory>
```

Example Command and Response The following command:

```
inventory.xml?action=getMoveResult&partition=partition+1
```

immediately returns the following XML-formatted data:

```
<inventory>
   <moveResult>
      Move barcode 345547L5 from Slot 4 to Slot 3 Successful
   </moveResult>
</inventory>
```

list

Lists all storage slots, entry/exit slots, and drives in the specified partition.

- For each slot and drive, the list indicates whether or not it is full.
- For each occupied slot or drive, the list also indicates the barcode information of the cartridge and whether or not the cartridge is queued for eject.

Syntax `inventory.xml?action=list&partition=[partition name]`

where the value for:

This parameter...	Specifies...
partition	The exact name of the partition for which you want a logical inventory list. **Notes:** ▪ The partition name is case-sensitive. ▪ Use the **partitionList.xml** command to retrieve a list of all the partitions currently configured in the library (see partitionList.xml on page 171). ▪ The partition name is set when the partition is created. Refer to your library *User Guide* for detailed information about configuring and using partitions in the library.

Command Response The command immediately returns the following XML-formatted data:

```
<inventory>
   <partition>
      <name>[value]</name>
      <storageSlot>
         <id>[first storage slot ID]</id>
         <offset>[value]</offset>
         <barcode>[value]</barcode>
         <isQueued>yes|no</isQueued>
         <full>yes|no</full>
      </storageSlot>
      ...
      <storageSlot>
         <id>[last storage slot ID]</id>
         <offset>[value]</offset>
         <barcode>[value]</barcode>
         <isQueued>yes|no</isQueued>
         <full>yes|no</full>
      </storageSlot>
```

```
            <entryExitSlot>
               <id>[first EE slot ID]</id>
               <offset>[value]</offset>
               <barcode>[value]</barcode>
               <isQueued>yes|no</isQueued>
               <full>yes|no</full>
            </entryExitSlot>
            ...
            <entryExitSlot>
               <id>[last EE slot ID]</id>
               <offset>[value]</offset>
               <barcode>[value]</barcode>
               <isQueued>yes|no</isQueued>
               <full>yes|no</full>
            </entryExitSlot>
            <drive>
               <id>[first drive]</id>
               <barcode>[value]</barcode>
               <isQueued>[value]</isQueued>
               <full>yes|no</full>
            </drive>
            ...
            <drive>
               <id>[last drive]</id>
               <offset>[value]</offset>
               <barcode>[value]</barcode>
               <sourceSlot>[value]</sourceSlot>
               <sourceOffset>[value]</sourceOffset>
               <isQueued>[value]</isQueued>
               <full>yes|no</full>
            </drive>
         </partition>
      </inventory>
```

where the value for:

This parameter...	Indicates...
name	The name of the partition.
storageSlot	That the following block of information is for a slot in the partition storage pool.
entryExitSlot	That the following block of information is for a slot in the partition entry/exit pool. **Notes:** ▪ For the T120 library, the number of entry/exit slots reported depends on the Eject Mode configured for the library. Refer to the *Spectra T120 Library User Guide* for information about eject modes. ▪ For all other libraries, This parameter is not returned if the partition does not have one or more chambers assigned to the entry/exit pool. Nor is returned if the entry/exit pool does not contain one or more magazines.
drive	That the following block of information is for a drive assigned to the partition.
id	The SCSI element address of the slot or drive that the library reports to the host.

This parameter...	Indicates...
offset	The BlueScale number assigned to the slot or drive in the specified pool. This number appears as the number for each slot or drive in the BlueScale Inventory screen. Refer to your library *User Guide* for information about the Inventory screen. **Note:** When creating a move queue file to be uploaded to the library, you can use the offset parameter values returned by the **inventory.xml** command as the *source_num* and *destination_num* parameters. See "Create a Move Queue File" in your library *User Guide* for detailed information about creating a move queue.
sourceSlot	Only returned in drive blocks. Matches with the value of the `<id>` field of the slot from which the tape in the drive came.
sourceOffset	Only returned in drive blocks. Matches with the value of the `<offset>` field of the slot from which the tape in the drive came.
barcode	The barcode label information of the cartridge in the slot or drive in the specified pool. **Note:** This parameter is only returned when the value for the **full** parameter for the slot or drive is **yes**.
isQueued	Whether the cartridge in the specified slot is queued for ejection from the library. Values: **yes, no** **Notes:** ▪ Queued ejects are only supported by the T120 library. For all other libraries, the value for this parameter is always **no**. ▪ This parameter is only returned when the value for the full parameter is **yes**.
cleaningCyclesTotal Cleaning cartridges only	The number of times the cartridge has been used to clean a tape drive.
cleaningCyclesLeft Cleaning cartridges only	The number of cleanings remaining.
full	Whether the slot or drive is full. Values: **yes**, **no**

Example Command and Response The following command:

```
inventory.xml?partition=Partition 1
```

retrieves the following cartridge inventory for a partition named Partition 1:

```
<inventory>
   <partition>
      <name>Partition 1</name>
      <storageSlot>
         <id>4096</id>
         <offset>1</offset>
         <barcode>83270L4</barcode>
         <isQueued>no</isQueued>
         <full>yes</full>
      </storageSlot>
      ...
      <storageSlot>
         <id>4151</id>
         <offset>47</offset>
         <full>no</full>
      </storageSlot>
      <entryExitSlot>
         <id>4625</id>
         <offset>52</offset>
         <full>no</full>
      </entryExitSlot>
      ...
      <drive>
         <id>256</id>
         <offset>1</offset>
         <barcode>519009L5 </barcode>
         <sourceSlot>4098</sourceSlot>
         <sourceOffset>3</sourceOffset>
         <isQueued>no</isQueued>
         <full>yes</full>
      </drive>
      <drive>
         <id>257</id>
         <offset>2</offset>
         <full>no</full>
      </drive>
   </partition>
</inventory>
```

move

Moves a tape from one location in the library to another.

Note: This action was added with BlueScale12.8.04.

Syntax `inventory.xml?action=move&partition=[partition name]&sourceID=[SLOT|EE|DRIVE|BC]&sourceNumber=[offset|barcode]&destinationID=[SLOT|EE|DRIVE]&destinationNumber=[offset]`

where the value for:

This parameter...	Specifies...
partition	The exact name of the partition for which you want to move tapes. **Notes:** ▪ The partition name is case-sensitive. ▪ Use the **partitionList.xml** command to retrieve a list of all the partitions currently configured in the library (see partitionList.xml on page 162). ▪ The partition name is set when the partition is created. Refer to your library *User Guide* for detailed information about configuring and using partitions in the library.
sourceID	The location type for the source from which you want to move the cartridge. Values: **SLOT** — The cartridge is moved from a slot in the storage pool. **EE** — The cartridge is moved from a slot in the entry/exit pool. **DRIVE** — The cartridge is moved from the drive. **BC** — The cartridge to be moved is identified by its barcode label information.
sourceNumber	Specifically identifies the cartridge to be moved, either by the ID of the slot or drive where it is currently located or by the barcode label on the cartridge. ▪ If the **sourceID** is **SLOT** or **EE**, this value is the number assigned to the slot where the cartridge is currently located. Locate the `<offset>` parameter for the desired cartridge in the result from the an `inventory.xml?action=list` command. ▪ If the **sourceID** is **DRIVE**, this value is the number of the drive where the cartridge is currently located. Locate the `<offset>` parameter for the drive in the result from the an `inventory.xml?action=list` command. ▪ If the **sourceID** is **BC**, this value is the human-readable barcode character string for the cartridge to be moved. Locate the `<barcode>` parameter for the desired cartridge in the result from the an `inventory.xml?action=list` command.
destinationID	The location type for the destination of the cartridge move. Values: ▪ **SLOT** — The cartridge is moved to an empty slot in the storage pool. ▪ **EE** — The cartridge is moved to an empty slot in the entry/exit pool. ▪ **DRIVE** — The cartridge is moved to the specified drive.
destinationNumber	Specifies the ID of the location to which the cartridge is moved. ▪ An empty slot in the entry/exit or storage pool for the partition. Locate the `<offset>` parameter for the desired slot in the result from the an `inventory.xml?action=list` command. ▪ The number of the drive. Locate the `<offset>` parameter for the drive in the result from the an `inventory.xml?action=list` command.

Command Response The command immediately returns the following XML-formatted data:

```
<inventory>
   <status>OK</status>
   <message>[message text]</message>
</inventory>
```

Progress Use `inventory.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 18), it is possible to issue another extended action command.

Use `inventory.xml?action=getMoveResult` to determine the results of the move. See getMoveResult for details.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>inventory.xml</line>
      <line>Query string:</line>
      <line>partition=[partition name]</line>
   </usage>
</syntaxError>
```

Example Command and Response The following command:

```
inventory.xml?action=move&partition=<Partition+1>&
sourceID=<BC>&sourceNumber=<83270L4>&
destinationID=<DRIVE>&destinationNumber=<1>
```

```
<inventory>
   <status>OK</status>
   <message>
      Started move. Set progress in your query for status.
   </message>
</inventory>
```

and moves the cartridge with barcode `83270L4` to drive 1.

CHAPTER 9

librarySettings

librarySettings.xml

Use the **librarySettings.xml** command to set or query selected library settings usually set on the System Configuration screen.

Note: This command was added with BlueScale12.6.45.

Action	
list	below
set	page 71

list

Description Returns a list of the current library settings.

Syntax `librarySettings.xml?action=list`

Command Response The command immediately returns the following XML-formatted data:

```
<librarySettings>
   <libraryName>[name in text]</libraryName>
   <autoLogoutTimeoutInMinutes>
      [number in minutes]
   </autoLogoutTimeoutInMinutes>
   <drivePerformanceMonitoringEnabled>
      yes|no
   </drivePerformanceMonitoringEnabled>
   <SNMPSettings>
      <enabled>yes|no</enabled>
      <systemContact>[contact string]</systemContact>
      <systemLocation>[location string]</systemLocation>
      <community>[community string]</community>
      ...
      <trapDestination>
         <community>[community string]</community>
         <description>[text]</description>
         <ipAddress>[valid IP address]</ipAddress>
      </trapDestination>
      ...
   </SNMPSettings>
   <automaticPowerUpAfterPowerFailureEnabled>
      yes|no
   </automaticPowerUpAfterPowerFailureEnabled>
</librarySettings>
```

where the value for:

This parameter...	Indicates...
libraryName	The name used to identify the library.
autoLogoutTimeout InMinutes	How long the current connection to the XML interface or BlueScale user interface can be idle before the current user is logged out. A value of 0 (zero) means the feature is disabled.
drivePerformance MonitoringEnabled	Whether the option to monitor the performance of the drives in the library is enabled (**yes**) or disabled (**no**). Values: **yes**, **no**
SNMPSettings	Lists whether the SNMP agent is enabled or disabled, and how the SNMP settings are configured. ▪ **enabled** - Whether the SNMP agent is enabled (**yes**) or disabled (**no**). Values: **yes**, **no** ▪ **systemContact** - Maps to the value for system.4 (sysContact) object in the MIB. ▪ **systemLocation** - Maps to the value for system.6 (sysLocation) object in the MIB. ▪ **community** - SNMP communities to which the library currently belongs. **Important:** Community strings are case sensitive. If the library is configured to include the community called "OurCommunity," it will answer to queries from OurCommunity but not from a community called "ourcommunity." ▪ **trapDestination** - the currently configured trap destinations (IP addresses) to which the library will send SNMP traps. **Note:** trapDestinations cannot be set using the XML interface. ▪ **community** = The community name for the trap destination. ▪ **description** = A description of the SNMP host or management system receiving the SNMP traps. ▪ **ipAddress** = The IP address of the SNMP host or management system receiving the SNMP traps.
automaticPowerUp AfterPowerFailure Enabled	Whether the option to automatically power on the library after a power failure is enabled (**yes**) or disabled (**no**). Values: **yes**, **no**

Example Command and Response The following command:

```
librarySettings.xml?action=list
```

immediately returns the following XML-formatted data:

```
<librarySettings>
   <libraryName>TFinityLibrary</libraryName>
   <autoLogoutTimeoutInMinutes>0</autoLogoutTimeoutInMinutes>
   <onlineAccessEnabled>no</onlineAccessEnabled>
   <drivePerformanceMonitoringEnabled>
      yes
   </drivePerformanceMonitoringEnabled>
   <SNMPSettings>
      <enabled>yes</enabled>
      <systemContact>User1</systemContact>
      <systemLocation>Tfinity Lab</systemLocation>
      <community>public</community>
      <community>private</community>
      <trapDestination>
         <community>public</community>
         <description>public</description>
         <ipAddress>1.1.1.1</ipAddress>
      </trapDestination>
      <trapDestination>
         <community>private</community>
         <description>private</description>
         <ipAddress>2.2.2.2</ipAddress>
      </trapDestination>
      <trapDestination>
         <community>test</community>
         <description>test</description>
         <ipAddress>3.3.3.3</ipAddress>
      </trapDestination>
   </SNMPSettings>
   <automaticPowerUpAfterPowerFailureEnabled>
      yes
   </automaticPowerUpAfterPowerFailureEnabled>
</librarySettings>
```

set

Description Configure a library setting usually configured from the BlueScale System Setup screen.

Note: Only one library setting can be set in each librarySettings command. To reset multiple parameters, you must issue separate commands.

Syntax `librarySettings.xml?action=set&libraryName=[library name text]&autoLogoutTimeoutInMinutes=[number in minutes]`
`&onlineAccess=[enable|disable]`
`&drivePerformanceMonitoring=[enable|disable]`
`&SNMPAgent=[enable|disable]`
`&automaticPowerUpAfterPowerFailure=[enable|disable]`

where the value for:

This parameter...	Specifies...
libraryName	The name used to identify the library.
autoLogoutTimeoutInMinutes	How long the current connection to the XML interface or BlueScale user interface can be idle before the current user is logged out. A value of 0 (zero) means the feature is disabled.
onlineAccessEnabled	Enabling or disabling links to the Spectra Logic website at the bottom of each BlueScale screen when the library has a connection to the Internet. **Note:** This parameter was removed in BlueScale12.7.04.02
drivePerformanceMonitoringEnabled	Enabling or disabling the option to monitor the performance of the drives in the library. Values: **enable**, **disable**
SNMPSettings	Enabling or disabling the SNMP agent. Values: **enable**, **disable** **Note:** SNMP settings cannot be edited using the XML interface.
automaticPowerUpAfterPowerFailureEnabled	Enabling or disabling the option to automatically power on the library after a power failure. Values: **enable**, **disable**

Command Response The command immediately returns the following XML-formatted data:

```
<librarySettings>
   <status>OK</status>
   <message><message text></message>
</librarySettings>
```

where the value for:

This parameter...	Indicates...
message	The system parameter that was set and to what it was set.

Example Command and Response The following command:

```
librarySettings.xml?action=set&SNMPAgent=enable
```

immediately returns the following XML-formatted data:

```
<librarySettings>
   <status>OK</status>
   <message>SNMPAgent set to enable</message>
</librarySettings>
```

CHAPTER 10

libraryStatus

libraryStatus.xml

Use the **libraryStatus.xml** command to get and refresh status information for the library.

Action	
[no parameters] or list	below
getMoveOperationDetails	page 91
RCMStatus	page 95
refreshECInfo	page 96
refreshEnvironment	page 97

[no parameters] or list

Description Returns the library type, serial number, component status, and engineering change level information for the library that received the command.

Syntax `libraryStatus.xml`

Command Response The command immediately returns the following XML-formatted data:

```
<libraryStatus>
   <libraryType>TFinity|T950|T680|T380|T200|T120</libraryType>
   <serialNumber>[value]</serialNumber>
   <libraryUpTimeSeconds>[value]</libraryUpTimeSeconds>
   <maintenanceMode>yes|no</maintenanceMode>
   <railPowerOn>yes|no</railPowerOn>
   <frames>
      <chassisCount>[value]</chassisCount>
      <physicalFrame>
         <framePosition>[value]<framePosition>
         <frameType>
            Left service frame|Right service frame
            Center TAP frame with drive frame id [value]|
            Drive expansion frame with drive frame id [value]|
            Media expansion frame|T680 frame|T380 frame|T200 frame
         <frameType>
      </physicalFrame>
      ...
   </frames>
```

```
<robot>
   <number>1|2</number>
   <state>inService|impaired|good|unknown</state>
   <transporterType>HPT|legacy</TransporterType>
   <serviceFrame>left|right|none|unknown</serviceFrame>
   <tapeInPickerCurrent>yes|no|unknown</tapeInPickerCurrent>
   <TeraPackInTransporterCurrent>
      yes|no|unknown
   </TeraPackInTransporterCurrent>
   <tapeInPickerUponService>
      yes|no|unknown
   </tapeInPickerUponService>
   <TeraPackInTransporterUponService>
      yes|no|unknown
   </TeraPackInTransporterUponService>
   <topHAXGear>Engaged|Not Engaged</topHAXGear>
   <bottomHAXSolenoid>Engaged|Not Engaged</bottomHAXSolenoid>
</robot>
...
<serialNumber>[value]</serialNumber>
<excessiveMoveFailures>
   <move>
      <partition>[value]</partition>
      <source>[value]</source>
      <destination>[value]</destination>
      <numberOfFailures>[value]</numberOfFailures>
      <lastSenseInfo>[value]</lastSenseInfo>
      <lastFailedMoveTime>[value]</lastFailedMoveTime>
   </move>
   ...
</excessiveMoveFailures>
<controllerEnvironmentInfo>
   <controller>
      <ID>[value]</ID>
      <temperatureInCelsius>[value]</temperatureInCelsius>
      <portALinkUp>yes|no</portALoopUp>
      <portBLinkUp>yes|no</portBLoopUp>
      <failoverStatus>
         Unconfigured|Unknown|Incompatible|Error|Passive|Active
      </failoverStatus>
   </controller>
   <driveControlModule>
      <ID>[value]</ID>
      <twelveVoltVoltage>[value]</twelveVoltVoltage>
      <fiveVoltVoltage>[value]</fiveVoltVoltage>
      <fanCurrentInAmps>[value]</fanCurrentInAmps>
      <temperatureInCelsius>[value]</temperatureInCelsius>
   </driveControlModule>
   ...
```

```
<powerSupplyFRU>
   <ID>[value]</ID>
   <inputPowerOkay>yes|no</inputPowerOkay>
   <outputPowerOkay>yes|no</outputPowerOkay>
   <temperatureWarning>yes|no</temperatureWarning>
   <temparatureAlarm>yes|no</temperatureAlarm>
   <modelNumber>[value]</modelNumber>
   <manufacturerPartNumber>[value]</manufacturerPartNumber>
   <serialNumber>[value]</serialNumber>
   <modLevel>[value]</modLevel>
   <manufacturer>[value]</manufacturer>
   <countryOfManufacturer>[value]</countryOfManufacturer>
   <temperatureInCelsius>[value]</temperatureInCelsius>
   <communicatingWithPCM>yes|no</communicatingWithPCM>
   <fanInPowerSupplyFRU>
      <number>1|2|3</number>
      <okay>yes</okay>
   </fanInPowerSupplyFRU>
   ...
   <powerSupplyInPowerSupplyFRU>
      <nominalVoltage>5|12|24</nominalVoltage>
      <actualVoltage>[value]</actualVoltage>
      <actualCurrentInAmps>[value]</actualCurrentInAmps>
   </powerSupplyInPowerSupplyFRU>
   ...
</powerSupplyFRU>
...
<powerControlModule>
   <ID>[value]</ID>
   <temperatureInCelsius>[value]</temperatureInCelsius>
   <parallelACPresent>yes|no</parallelACPresent>
   <primaryACPresent>yes|no</primaryACPresent>
   <secondaryACPresent>yes|no</secondaryACPresent>
   <supplyDetectionWorking>yes|no</supplyDetectionWorking>
   <ACCurrentInAmps>[value]</ACCurrentInAmps>
   <primaryACVoltage>[value]</primaryACVoltage>
   <secondaryACVoltage>[value]</secondaryACVoltage>
   <twelveVoltVoltage>[value]</twelveVoltVoltage>
   <fiveVoltVoltage>[value]</fiveVoltVoltage>
   <onBoardTemperatureInCelsius>[value]
      </onBoardTemperatureInCelsius>
   <remoteTemperatureInCelsius>[value]
      </remoteTemperatureInCelsius>
   <powerSupplyInPCM>
      <position>[value]</position>
      <faulted>yes|no</faulted>
   </powerSupplyInPCM>
   ...
</powerControlModule>
```

```
<fanControlModule>
   <ID>[value]</ID>
   <frameNumber>[value]</frameNumber>
   <temperatureInCelsius>[value]</temperatureInCelsius>
   <backPanelSwitch>open|closed</backPanelSwitch>
   <fanPanelSwitch>open|closed</fanPanelSwitch>
   <filterPanelSwitch>open|closed</filterPanelSwitch>
   <frontTAPFramePanelSwitch>
      open|closed
   </frontTAPFramePanelSwitch>
   <boardVoltage>[value]</boardVoltage>
   <fanInputVoltage>[value]</fanInputVoltage>
   <fanSpeedVoltage>[value]</fanSpeedVoltage>
   <fanSpeedSetting>[value]</fanSpeedSetting>
   <newFansCalibrated>yes|no</newFansCalibrated>
   <newFilterCalibrated>yes|no</newFilterCalibrated>
   <fanInFCM>
      <number>1|2|3|4|5|6|7|8|9|10</number>
      <on>yes|no</on>
      <speedInRPM>[value]</speedInRPM>
   </fanInFCM>
   ...
   <lightBank>
      <number>1|2|3</number>
      <on>yes|no</on>
   </lightBank>
   ...
</fanControlModule >
<frameManagementModule>
   <ID>[value]</ID>
   <twentyFourVoltVoltage>[value]</twentyFourVoltVoltage>
   <fiveVoltVoltage>[value]</fiveVoltVoltage>
   <fanRailVoltage>[value]</fanRailVoltage>
   <switchedRailVoltage>[value]</switchedRailVoltage>
   <twentyFourVoltCurrentInAmps>[value]
      </twentyFourVoltCurrentInAmps>
   <powerConsumedInWatts>[value]</powerConsumedInWatts>
   <sampleRateInSeconds>[value]</sampleRateInSeconds>
   <samplesTaken>[value]</samplesTaken>
   <temperatureInCelsius>[value]</temperatureInCelsius>
   <EPMTemperatureInCelsius>[value]</EPMTemperatureInCelsius>
   <frameToFrameTemperatureInCelsius>
      [value]
   </frameToFrameTemperatureInCelsius>
   <frameToFrameAttached>yes|no</frameToFrameAttached>
   <frameToFrameFiveVoltEnabled>
      yes|no
   </frameToFrameFiveVoltEnabled>
   <fansEnabled>yes|no</fansEnabled>
   <backSwitchOpen>yes|no</backSwitchOpen>
   <filterSwitchOpen>yes|no</filterSwitchOpen>
   <frontSwitchOpen>yes|no</frontSwitchOpen>
   <safetyInterlockOpen>yes|no</safetyInterlockOpen>
   <frameIDInfo>[value]</frameIDInfo>
   <driveFrameNumber>[value]</driveFrameNumber>
```

```
            <switchedRailState>
               ground|24 volts|neither
            </switchedRailState>
            <robotPowerEnabled>yes|no</robotPowerEnabled>
            <internalLightsEnabled>yes|no</internalLightsEnabled>
            <externalLightsEnabled>yes|no</externalLightsEnabled>
            <fanPair>
               <number>1|3|5|7|9</number>
               <present>yes|no</present>
            </fanPair>
            ...
            <fanInFMM>
               <number>1|2|3|4|5|6|7|8|9|10</number>
               <on>yes|no</on>
               <speedInRPM>[value]</speedInRPM>
            </fanInFMM>
            ...
            <powerSupplyInFMM>
               <position>[value]</position>
               <faulted>yes|no</faulted>
            </powerSupplyInFMM>
            ...
         </frameManagementModule >
         ...
         <serviceBayControlModule>
            <ID>[value]</ID>
            <frameIDInfo>[value]</frameIDInfo>
            <safetyDoorState>[value]</safetyDoorState>
            <overrideSwitch>[value]</overrideSwitch>
            <rearAccessPanel>open|closed</rearAccessPanel>
            <sideAccessPanel>open|closed</sideAccessPanel>
            <sidePanel>open|closed</sidePanel>
            <robotInServiceFrame>yes|no</robotInServiceFrame>
            <bulkIEPresent>yes|no</bulkIEPresent>
            <bulkIEDoorOpen>yes|no</bulkIEDoorOpen>
            <bulkIEAjar>yes|no</bulkIEAjar>
            <solenoidPinPosition>
               fully extended|undetermined
            </solenoidPinPosition>
            <bulkTAPLocation>left|right</bulkTAPLocation>
         </serviceBayControlModule>
      </controllerEnvironmentInfo>

      <ECInfo>
         <component>
            <ID>[value]</ID>
            <EC>[value]</EC>
            <serialNumber>[value]</serialNumber>
            <topLevelAssemblyEC>[value]</topLevelAssemblyEC>
            <topLevelAssemblySerialNumber>[value]
               </topLevelAssemblySerialNumber>
            <date>[value]</date>
         </component>
         ...
      </ECInfo>
```

```
   <SDInfo>
      <freeSpaceInMB>[value]</freeSpaceInMB>
   </SDInfo>
</libraryStatus>
```

where the value for:

This parameter...	Indicates...
libraryType	The type of Spectra Logic library that received the command. Values: **TFinity**, **T950**, **T680**, **T380**, **T200**, or **T120**
serialNumber	The library serial number (hardware ID).
libraryUpTime Seconds	The time in seconds since the library was powered on.
maintenanceMode	Whether the library is in Maintenance Mode. Values: **yes**, **no**
frames	Lists information about the frames in the library. Values: ▪ **chassisCount** - the number of frames in the library. **Note:** FOr T680, T380, and T200 libraries, **chassisCount** will always be 1. ▪ **physicalFrame** - information about each frame in the library. ▪ **framePosition** - The position of the frame, from left to right as viewed from the front. ▪ **frameType** - The frame type. Values: **Left service frame**, **Right service frame**, **Center TAP frame with drive frame id [*value*]**, **Drive expansion frame with drive frame id [*value*]**, **Media expansion frame**, **T680 frame**, **T380 frame**, **T200 frame**
railPowerOn TFinity libraries only	Whether the rail power is on (yes) or off (no). Values: **yes, no**

This parameter...	Indicates...
robot ▪ Not returned for T50e and T120 libraries ▪ Only **state is** returned for T200, T380, T680, and T950 libraries ▪ All sections returned for TFinity libraries	The status of the robot(s). There is one section for each robot . **Note:** If the LCM does not report any robots or has an error accessing the robot data, the robot parameter is not included. Values: ▪ **number** - The number of the robot in the library. Numbering is from left to right as viewed from the front of the library. Values: **1** (left robot) or **2** (right robot). ▪ **state** - The status of the robot. Values: **inService**, **impaired**, **good**, or **unknown** If **state = inService** then the following transporter state is reported: ▪ tapeInPickerCurrent = Whether the picker is currently holding a tape. Values: **yes**, **no**, **unknown** (uncommon) ▪ TeraPackInTransporterCurrent = Whether the transporter is currently holding a TeraPack magazine. Values: **yes**, **no**, **unknown** (uncommon) ▪ tapeInPickerUponService = Whether there was a tape in the picker when the robot went into service. Values: **yes**, **no**, **unknown** (uncommon) ▪ TeraPackInTransporterUponService = Whether there was a TeraPack magazine in the transporter when the robot went into service. Values: **yes**, **no**, **unknown** (uncommon) ▪ **transporterType -** The type of transporter. Values: **HPT** (high performance transporter), **legacy** (original transporter) **serviceFrame** - The service frame containing an "in service" robot. Values: **left**, **right**, **none** (uncommon), or **unknown** (uncommon) **Note:** The serviceFrame tag is only shown if **state = inService**. ▪ **topHAXGear** - The state of the HAX gear at the top of the TeraPorter. Values: **Engaged, Not Engaged** ▪ **bottomHAXSolenoid** - The state of the HAX solenoid. Values: **Engaged, Not Engaged**

This parameter...	Indicates...
excessiveMove Failures	Lists information for the last unique moves, up to ten, that failed five consecutive times. ▪ **move** - Information for one unique move. ▪ **partition** = The name of the partition in which the move was attempted. ▪ **source** = The element type and BlueScale number assigned to the source of the move. This number appears as the number for each slot or drive in the BlueScale Inventory screen. ▪ **destination** = The element type and BlueScale number assigned to the destination of the move. This number appears as the number for each slot or drive in the BlueScale Inventory screen. ▪ **numberOfFailures** = The number of consecutive failures recorded for this move. ▪ **lastSenseInfo** = The Sense Key, Additional Sense Code (ASC), and Additional Sense Code Qualifiers (ASQC) for the move, in the format 0x*nn*, 0x*nn*, 0x*nn*. **Note:** See *Spectra Tape Libraries SCSI Developer's Guide* for sense code information. ▪ **lastFailedMoveTime** = Timestamp of the last failed move in the format "YYYY/MM/DD HH:MM:SS"
controller EnvironmentInfo	The status of different components in the library. The information provided for each component is described below.
	controller - QIPs and RIMs ▪ **ID** - The component identifier for the exporting RIM or F-QIP (primary controller) using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the controller. Not used in T120 library component identifiers. ▪ **F-QIP[*integer*]** = The designator of the controller bay where the QIP or RIM is installed. For all libraries except the T120, the value of [***integer***] is always **1**. For the T120 library, the value of [***integer***] is either **1** or **2**. ▪ **temperatureInCelsius** - The measured temperature in Celsius. ▪ **portALinkUp** - Whether or not there is an active fibre connection on port A. Values: **yes, no** ▪ **portBLinkUp** - Whether or not there is an active fibre connection on port B. Values: **yes, no** ▪ **failoverStatus** - Whether controller failover is configured/active for the controller. Values: **Unconfigured**, **Unknown**, **Incompatible**, **Error**, **Passive**, **Active**

This parameter...	Indicates...
controller EnvironmentInfo (*continued*)	**driveControlModule** - The drive sled that houses the drive, providing it power and a location based identifier. ▪ **ID** - The component identifier for the drive using the form **FR[*integer*]/DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the controller. Not used in T120 library component identifiers. ▪ **[*interface*]** = The interface used by the drive. Values: ▪ **f** = Fibre Channel ▪ **s** = Serial Attached SCSI (SAS) ▪ (blank) = SCSI ▪ **[*technology*]** = The technology used by the drive. Values: **LTO**, **TS11x0, T10K** ▪ **DRV[*integer*]** = The designator of the drive bay in the DBA, as viewed from the back of the library. For all libraries except the T120, the value of ***x*** can be **1** through **4**. For the T120 library, the value of ***x*** can be **1** through **6** for full-height drives, and **1b** through **6a** for half-height drives. ▪ **twelveVoltVoltage** - The measured voltage of the 12 volt power supply. ▪ **fiveVoltVoltage** - The measured voltage of the 5 volt power supply. ▪ **fanCurrentInAmps** - The measured current being drawn by the drive sled fan. ▪ **temperatureInCelsius** - The measured temperature in Celsius.

This parameter...	Indicates...

controller EnvironmentInfo (*continued*)	**powerSupplyFRU** - The 5/12 VDC and 24 VDC power supply modules convert AC input to provide the 5 VDC and 12 VDC power used by the drives in the frame and the 24 VDC required by the robotics.
	▪ **ID** - The component identifier for the power module using the form **FR[*integer*]/PCM/PowerSupply[*integer*]** where:
	  ▪ **FR[*integer*]**= The number of the frame. Only used in the component identifier when the module is in a library that supports multiple frames.
	  ▪ **PCM** = Power Control Module.
	  ▪ **PowerSupply[*integer*]** = The number of the power supply module in the PCM. Values: **1**, **2**, **3**, **4**, **5**, **6**, **7** (5/12 volt supplies), **8**, **9** (24 volt supplies)
	▪ **inputPowerOkay** - Whether the input voltages to the power supply module are within normal ranges. Values: **yes, no**
	▪ **outputPowerOkay** - Whether the output voltage(s) from the power supply module are within normal ranges. Values: **yes, no**
	▪ **temperatureWarning** - Whether the power supply module has indicated a temperature warning (greater than 140° F (60° C)). Values: **yes, no**
	▪ **temperatureAlarm** - Whether the power supply module has indicated a temperature alarm (greater than 147° F (64° C)). Values: **yes, no**
	▪ **modelNumber** - The model number of the part.
	▪ **manufacturerPartNumber** - The manufacturer's part number.
	▪ **serialNumber** - The serial number of the part.
	▪ **modLevel** - The modification level for the part.
	▪ **manufacturer** - The manufacturer of the part.
	▪ **countryOfManufacturer** - The country where the part was manufactured.
	▪ **temperatureInCelsius** - The measured temperature in Celsius.
	▪ **communicatingWithPCM** - Whether the power supply module is communicating with the power control module. Values: **yes, no**
	▪ **fanInPowerSupply** - Information for each fan in the module.
	  ▪ **number** = the number of the fan. Values: **1**, **2**, **3**
	  ▪ **okay** = Whether the fan is operating correctly. Values: **yes, no**
	▪ **powerSupplyInPowerSupplyFRU** - Information for each power supply.
	  ▪ **nominalVoltage** = The expected output from the power supply. Values **5**, **12**, **24**
	  ▪ **actualVoltage** = The measured output from the power supply.
	  ▪ **actualCurrentInAmps** = The measured current in Amps (not applicable for 24 volt supplies).

This parameter...	Indicates...

controller EnvironmentInfo (*continued*)	**powerControlModule** - The power control module can be configured to supply Dual or Parallel AC power to the power supplies. Along with supplying power to the power supplies, the PCM monitors output voltage, output current, and PCM operational temperature.
	▪ **ID** - The component identifier for the power module using the form **FR[*integer*]/PCM** where:
	  ▪ **FR[*integer*]** = The designator of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames.
	  ▪ **PCM** = Power Control Module.
	▪ **parallelACPresent** - Whether the library has an optional parallel AC power module. This module provides AC power to two separate banks of 5/12 volt and 24 volt supplies. Values: **yes, no**
	▪ **primaryACPresent** - Whether power is connected to the primary AC input. Values: **yes, no**
	▪ **secondaryACPresent** - Whether power is connected to the secondary AC input. Values: **yes, no**
	▪ **supplyDetectionWorking** - Indicates if the library is able to communicate with the PCM IO expander. Values: **yes, no**
	▪ **ACCurrentInAmps** - This is no longer used. It always shows a value of **0**.
	▪ **primaryACVoltage** - The measured primary AC voltage input.
	▪ **secondaryACVoltage** - The measured secondary AC voltage input.
	▪ **twelveVoltVoltage** - The measured 12 volt supply voltage.
	▪ **fiveVoltVoltage** - The measured 5 volt supply voltage.
	▪ **onBoardTemperatureInCelsius** - The temperature, in Celsius, measured at the circuit board in the PCM.
	▪ **remoteTemperatureInCelsius** - This is no longer used. It always shows a value of **0**.
	▪ **powerSupplyInPCM** - Information for each power supply in the module.
	  ▪ **position** = The power supply location in the PCM. Values: **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**
	  ▪ **faulted** = Whether the power supply is faulted. Values: **yes, no**

This parameter...	Indicates...

controller EnvironmentInfo (*continued*)	**fanControlModule** - The fan control module controls the fan power and fan speed for the fans in each frame that provide airflow to keep the frame at a consistent operating temperature. The FCM also controls the lights in T200, T380, T680, and T950 (not including T950B) libraries.
	▪ **ID** - The component identifier for the module using the form **FR[*integer*]/FCM** where:
	  ▪ **FR[*integer*]** = The designator of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames.
	  ▪ **FCM** = Fan Control Module.
	▪ **temperatureInCelsius** - The measured temperature in Celsius.
	▪ **backPanelSwitch** - Whether the back panel switch is closed or open. Values: **closed**, **open**
	▪ **fanPanelSwitch** - Whether the fan panel switch is closed or open. Values: **closed**, **open**
	▪ **filterPanelSwitch** - Whether the filter panel switch is closed or open. Values: **closed**, **open**
	▪ **frontTAPFramePanelSwitch** - Whether the front TAP frame panel switch is closed or open. Values: **closed**, **open**
	▪ **boardVoltage** - The measured value of the 24 volt supply on the FCM.
	▪ **fanInputVoltage** - The measured fan input voltage.
	▪ **fanSpeedVoltage** - The measured fan speed voltage.
	▪ **fanSpeedSetting** - The set fan speed.
	▪ **newFansCalibrated** - This is no longer used. It always shows a value of **no**.
	▪ **newFilterCalibrated** - This is no longer used. It always shows a value of **no**.
	▪ **fanInFCM** - Information for each fan controlled by the fan control module.
	  ▪ **number** = The number of the fan. Values: **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**
	  ▪ **on** = Whether the fan is on (yes) or off (no). Values: **yes**, **no**
	  ▪ **speedInRPM** = The measured speed of the fan.
	▪ **lightBank** - Information for each light bank controlled by the fan control module.
	  ▪ **number** = The number of the light bank. Values: **1**, **2**, **3**
	  ▪ **on** = Whether the light bank is on (yes) or off (no). Values: **yes**, **no**

This parameter...	Indicates...

controller EnvironmentInfo (*continued*)	**frameManagementModule** - The frame management module, located in the bottom front of each T950B and TFinity frame, stores frame ID information and controls miscellaneous items like fans and lights.
	▪ **ID** - The component identifier for the module using the form **FR[*integer*]/FMM** where:
	  ▪ **FR[*integer*]** = The designator of the frame.
	  ▪ **FMM** = Frame Management Module.
	▪ **twentyFourVoltVoltage** - The measured voltage of the 24 volt input.
	▪ **fiveVoltVoltage** - The measured voltage of the 5 volt input.
	▪ **fanRailVoltage** - The measured fan rail voltage.
	▪ **switchedRailVoltage** - The measured switched rail voltage.
	▪ **twentyFourVoltCurrentInAmps** - The measured current for the 24 volt input.
	▪ **powerConsumedInWatts** - The measured power consumed.
	▪ **sampleRateInSeconds** - The sample rate in seconds.
	▪ **samplesTaken** - The number of samples taken.
	▪ **temperatureInCelsius** - The measured temperature in Celsius.
	▪ **EPMTemperatureInCelsius** - The measured temperature of the service frame power module in Celsius. TFinity libraries only.
	▪ **frameToFrameTemperatureInCelsius** - The measured temperature of the frame to frame board in Celsius.
	▪ **frameToFrameAttached** - Whether the frame to frame board cables are attached. Values: **yes**, **no**
	▪ **frameToFrame5VoltEnabled** - Whether the frame to frame board 5 volt supply is enabled. Values: **yes**, **no**
	▪ **fansEnabled** - Whether the frame management module fans are enabled. Values: **yes**, **no**
	▪ **backSwitchOpen** - Whether the frames back interlock switch is open. Values: **yes**, **no**
	▪ **filterSwitchOpen** - Whether the frames filter interlock switch is open. Values: **yes**, **no**
	▪ **frontSwitchOpen** - Whether the frames front interlock switch is open. Values: **yes**, **no**
	▪ **safetyInterlockOpen** - Whether the frames safety interlock switch is open. Values: **yes**, **no**
	▪ **frameIDInfo** - Frame numbers start with 80 for the main frame and are numbered in hexadecimal, counting down to the left with the next frame being 7f, and counting up to the right with the next frame being 81.
	▪ **driveFrameNumber** - The value of the drive frame switch (0-4), which is used to set the CAN addresses of all QIPs and drives. The main frame is always drive frame 0. Not applicable for media, service or bulk TAP frames.
	▪ **switchedRailState** - The setting for the fixed rail. Values: **ground**, **24 volts**, **neither**
	▪ **robotPowerEnabled** - Whether the robot has power enabled. Values: **yes, no**
	▪ **internalLightsEnabled** - Whether the internal lights are enabled. Values: **yes**, **no**
	▪ **externalLightsEnabled** - Whether the external lights are enabled. TFinity libraries only. Values: **yes**, **no**

This parameter...	Indicates...

controller EnvironmentInfo (*continued*)	**frameManagementModule** (*continued*)
	▪ **fanPair** - Information for each fan pair controlled by the frame management module.
	  ▪ **number** = The number for the fan pair. Values: **1**, **3**, **5**, **7**, **9**
	  ▪ **present** = Whether the fan is present. Values: **yes**, **no**
	▪ **fanInFMM** - Information for each fan in the fan pair.
	  ▪ **number** = The number for the fan. Values **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **10**
	  ▪ **on** = Whether the fan is powered on. Values: **yes**, **no**
	  ▪ **speedInRPM** = The measured speed of the fan.

	serviceBayControlModule - The service bay control module, located in each TFinity service frame, controls the service frame interlocks and provides bulk TAP status, if present. This is only applicable for TFinity libraries.
	▪ **ID** - The component identifier for the module using the form **FR[*integer*]/SCM** where:
	  ▪ **FR[*integer*]** = The designator of the frame.
	  ▪ **SCM** = Service Control Module.
	▪ **frameIDInfo** - Frame numbers start with 80 for the main frame and are numbered in hexadecimal, counting down to the left with the next frame being 7f, and counting up to the right with the next frame being 81.
	▪ **safetyDoorState** - The state of the service frame safety door. The manually operated service bay safety door slides closed to isolate the TeraPorter in the service bay, making it possible to continue library operations while you service the TeraPorter.
	Values: **open**, **closed**
	▪ **overrideSwitch** - The state of the service frame override switch.
	Values: **active**, **inactive**
	▪ **rearAccessPanel** - The state of the rear access panel interlock switch.
	Values: **open**, **closed**
	▪ **sideAccessPanel** - The state of the side access panel interlock switch.
	Values: **open**, **closed**
	▪ **sidePanel** - The state of the side panel interlock switch. Values: **closed**, **open**
	▪ **robotInServiceFrame** - Whether the robot is in the service position in the service frame. Values: **yes**, **no**
	▪ **bulkIEPresent** - Whether the service frame includes a bulk TAP. Values: **yes**, **no**
	▪ **bulkIEDoorOpen** - Whether the bulk TAP door is open (neither latch engaged). Values: **yes**, **no**
	▪ **bulkIEAjar** - Whether the bulk TAP door is ajar (only one latch engaged).
	Values: **yes**, **no**
	▪ **solenoidPinPosition** - Whether the bulk TAP carousel solenoid pin is fully extended. This is an indication of whether the carousel is fully rotated, either facing into the library or facing out toward the door, or in an undetermined position. Values: **fully extended**, **undetermined**
	▪ **bulkTAPLocation** - If **bulkIEPresent** is yes, **bulkTAPLocation** indicates the location of the bulk TAP, as viewed from the front. Values: **left**, **right**

This parameter...	Indicates...
controller EnvironmentInfo (*continued*)	**ECInfo** - The engineering change level for library components. **Note:** For components with a drive ID (for example, DBA1/fLTO-DRV2), the information is for the drive sled (DCM) and not the drive itself. ▪ **component** - Information for a specific component. ▪ **ID** = A description of the component including the frame number, DBA number, and component number, if relevant. ▪ **serialNumber** = The serial number of the component, if relevant. ▪ **topLevelAssemblyEC** = The EC level of the top level assembly. For example, the EC level of the printed circuit board in the RCM. ▪ **topLevelAssemblySerialNumber** = The serial number of the top level assembly, if relevant. ▪ **date** = The date of manufacture.
SDInfo	Lists information about the SD card in the LCM. **freeSpaceInMB** - Gives the amount of free space on the SD card in MB.

Example Command and Response The following command:

```
libraryStatus.xml
```

immediately returns the following XML-formatted data when issued to a TFinity library:

```
<libraryStatus>
   <libraryType>TFinity</libraryType>
   <serialNumber>1129A02</serialNumber>
   <libraryUpTimeSeconds>4334</libraryUpTimeSeconds>
   <maintenanceMode>no</maintenanceMode>
   <frames>
      <chassisCount>5</chassisCount>
      <physicalFrame>
         <framePosition>1</framePosition>
         <frameType>Left service frame</frameType>
      </physicalFrame>
      ...
   </frames>
   <railPowerOn>yes</railPowerOn>
   <robot>
      <number>1</number>
      <state>good</state>
      <transporterType>Legacy</transporterType>
      <topHAXGear>Engaged</topHAXGear>
      <bottomHAXGear>Engaged</bottomHAXGear>
      <topHAXSolenoid>Engaged</topHAXSolenoid>
      <bottomHAXSolenoid>Engaged</bottomHAXSolenoid>
   </robot>
   ...
```

```
<controllerEnvironmentInfo>
   <controller>
      <ID>FR3/DBA5/F-QIP1</ID>
      <temperatureInCelsius>21</temperatureInCelsius>
      <portALinkUp>yes</portALinkUp>
      <portBLinkUp>no</portBLinkUp>
      <failoverStatus>Unconfigured</failoverStatus>
      <type>8-Gbps FC RIM2</type>
      <fwVersion>8.3.0.1</fwVersion>
   </controller>
   <driveControlModule>
      <ID>FR3/DBA4/fLTO-DRV3</ID>
      <twelveVoltVoltage>12.000</twelveVoltVoltage>
      <fiveVoltVoltage>5.075</fiveVoltVoltage>
      <fanCurrentInAmps>0.136</fanCurrentInAmps>
      <temperatureInCelsius>31</temperatureInCelsius>
      <isFullHeight>yes</isFullHeight>
      <type>IBM Ultrium-TD6 Fibre</type>
      <driveFwVersion>JAX0</driveFwVersion>
   </driveControlModule>
   ...
   <powerSupplyFRU>
      <ID>FR3/PCM/PowerSupply1</ID>
      <inputPowerOkay>yes</inputPowerOkay>
      <outputPowerOkay>yes</outputPowerOkay>
      <temperatureWarning>no</temperatureWarning>
      <temperatureAlarm>no</temperatureAlarm>
      <modelNumber>SP525-Y02A</modelNumber>
      <manufacturerPartNumber>
         SP525-Y02A
      </manufacturerPartNumber>
      <serialNumber>BM22723</serialNumber>
      <modLevel>A</modLevel>
      <manufacturer>CHEROKEE INTL</manufacturer>
      <countryOfManufacturer>CHINA</countryOfManufacturer>
      <temperatureInCelsius>10</temperatureInCelsius>
      <communicatingWithPCM>yes</communicatingWithPCM>
      <fanInPowerSupplyFRU>
         <number>1</number>
         <okay>yes</okay>
      </fanInPowerSupplyFRU>
      ...
      <powerSupplyInPowerSupplyFRU>
         <nominalVoltage>12.000</nominalVoltage>
         <actualVoltage>12.171</actualVoltage>
         <actualCurrentInAmps>1.021</actualCurrentInAmps>
      </powerSupplyInPowerSupplyFRU>
      ...
   </powerSupplyFRU>
   ...
```

```
<powerControlModule>
   <ID>FR3/PCM</ID>
   <parallelACPresent>no</parallelACPresent>
   <primaryACPresent>no</primaryACPresent>
   <secondaryACPresent>no</secondaryACPresent>
   <supplyDetectionWorking>yes</supplyDetectionWorking>
   <ACCurrentInAmps>0</ACCurrentInAmps>
   <primaryACVoltage>0</primaryACVoltage>
   <secondaryACVoltage>0</secondaryACVoltage>
   <twelveVoltVoltage>0.180</twelveVoltVoltage>
   <fiveVoltVoltage>5.068</fiveVoltVoltage>
   <onBoardTemperatureInCelsius>
      37
   </onBoardTemperatureInCelsius>
   <remoteTemperatureInCelsius>
      0
   </remoteTemperatureInCelsius>
   <powerSupplyInPCM>
      <position>1</position>
      <faulted>no</faulted>
   </powerSupplyInPCM>
   ...
</powerControlModule>
...
```

```
<frameManagementModule>
   <ID>FR1/FMM</ID>
   <frameIDInfo>0x7e</frameIDInfo>
   <twentyFourVoltVoltage>23.657</twentyFourVoltVoltage>
   <fiveVoltVoltage>5.076</fiveVoltVoltage>
   <fanRailVoltage>13.792</fanRailVoltage>
   <switchedRailVoltage>23.872</switchedRailVoltage>
   <twentyFourVoltCurrentInAmps>
      0.000
   </twentyFourVoltCurrentInAmps>
   <powerConsumedInWatts>0</powerConsumedInWatts>
   <sampleRateInSeconds>2</sampleRateInSeconds>
   <samplesTaken>29</samplesTaken>
   <temperatureInCelsius>24</temperatureInCelsius>
   <EPMTemperatureInCelsius>44</EPMTemperatureInCelsius>
   <frameToFrameTemperatureInCelsius>
      0
   </frameToFrameTemperatureInCelsius>
   <frameToFrameAttached>yes</frameToFrameAttached>
   <frameToFrame5VoltEnabled>yes</frameToFrame5VoltEnabled>
   <fansEnabled>yes</fansEnabled>
   <backSwitchOpen>yes</backSwitchOpen>
   <filterSwitchOpen>no</filterSwitchOpen>
   <frontSwitchOpen>no</frontSwitchOpen>
   <safetyInterlockOpen>no</safetyInterlockOpen>
   <driveFrameNumber>0x2</driveFrameNumber>
   <switchedRailState>24Volt</switchedRailState>
   <robotPowerEnabled>no</robotPowerEnabled>
   <internalLightsEnabled>no</internalLightsEnabled>
   <externalLightsEnabled>no</externalLightsEnabled>
   <fanPair>
      <number>1</number>
      <present>yes</present>
   </fanPair>
   <fanPair>
      <number>2</number>
      <present>no</present>
   </fanPair>
   <fanInFMM>
      <number>1</number>
      <speedInRPM>7752</speedInRPM>
   </fanInFMM>
   ...
</frameManagementModule>
...
```

```
            <serviceBayControlModule>
               <ID>FR1/SCM</ID>
               <frameIDInfo>0x7e</frameIDInfo>
               <safetyDoorState>Open and locked</safetyDoorState>
               <overrideSwitch>inactive</overrideSwitch>
               <rearAccessPanel>closed</rearAccessPanel>
               <sideAccessPanel>closed</sideAccessPanel>
               <sidePanel>closed</sidePanel>
               <robotInServiceFrame>no</robotInServiceFrame>
               <bulkIEPresent>no</bulkIEPresent>
               <bulkIEDoorOpen>no</bulkIEDoorOpen>
               <bulkIEAjar>yes</bulkIEAjar>
               <solenoidPinPosition>undetermined</solenoidPinPosition>
               <bulkTAPLocation>left</bulkTAPLocation>
            </serviceBayControlModule>
            ...
         </controllerEnvironmentInfo>
         <ECInfo>
            <component>
               <ID>LCM Spectra PC</ID>
               <EC>10</EC>
               <serialNumber>121712-064</serialNumber>
               <topLevelAssemblyEC>11</topLevelAssemblyEC>
               <topLevelAssemblySerialNumber>
                  LS31320098
               <topLevelAssemblySerialNumber>
               <date>03/08/2018</date>
               <isGen3>yes</isGen3>
            </component>
            ...
         </ECInfo>
         <SDInfo>
            <freeSpaceInMB>405.529</freeSpaceInMB>
         </SDInfo>
      </libraryStatus>
```

getMove Operation Details

Description Displays operational details for the last ten completed moves including source and destination information, the start and stop time, overall move status, and the last operation of each of the two movers with respect to the move.

Notes:
- This action is only supported on TFinity libraries.
- This action was added with BlueScale12.7.00.

Syntax `libraryStatus.xml?action=getMoveOperationDetails`

Command Response The command immediately returns the following XML-formatted data:

```
<libraryStatus>
   <moveOperationDetails>
      <move>
         <startTime>[value]</startTime>
         <endTime>[value]</endTime>
         <overallStatus>success|fail</overallStatus>
         <overallSense>[value]</overallSense>
         <source>
            <partition>Partition Name|none</partition>
            <elementType>storage|IE|drive|other</elementType>
            <elementOffset>[value]</elementOffset>
         </source>
         <destination>
            <partition>Partition Name|none</partition>
            <elementType>storage|IE|drive|other</elementType>
            <elementOffset>[value]</elementOffset>
         </destination>
         <firstRobot>
            <number>1|2</number>
            <lastOperation>pick tape from slot|put tape to slot
               |pick magazine from drawer|put magazine to drawer
               |pick tape from drive|put tape to drive|other
            </lastOperation>
         </firstRobot>
         <secondRobot>
            <number>1|2</number>
            <lastOperation>pick tape from slot|put tape to slot
               |pick magazine from drawer|put magazine to drawer
               |pick tape from drive|put tape to drive|other
            </lastOperation>
         </secondRobot>
      </move>
      ...
   </moveOperationDetails>
</libraryStatus>
```

where the value for:

This parameter...	Indicates...
move	The container for information about one move.
startTime	The start time of the move in the format YYYY/MM/DD HH:MM:SS.
endTime	The end time of the move in the format YYYY/MM/DD HH:MM:SS.
overallStatus	The overall status of the move. Values: **success**, **fail**
overallSense	The Sense Key, Additional Sense Code (ASC), and Additional Sense Code Qualifiers (ASQC) for the move, in the format 0x*nn*, 0x*nn*, 0x*nn*. See *Spectra Tape Libraries SCSI Developer's Guide* for sense code information.

This parameter...	Indicates...

source	Information about the source for the move.
	▪ **partition** - The name of the partition containing the source, if applicable, or **none** if not applicable.
	▪ **elementType** - the type of element in which the source is contained. Values:
	  ▪ **storage** - The storage pool for the partition.
	  ▪ **IE** - The Entry/Exit pool of the partition.
	  ▪ **drive** - A drive assigned to the partition.
	  ▪ **other** - Some other storage element.
	elementOffset - The BlueScale number assigned to the slot or drive in the specified pool. This number appears as the number for each slot or drive in the BlueScale Inventory screen. Refer to your library *User Guide* for information about the Inventory screen.

destination	Information about the destination for the move.
	▪ **partition** - The name of the partition containing the destination, if applicable, or **none** if not applicable.
	▪ **elementType** - the type of element in which the destination is contained. Values:
	  ▪ **storage** - The storage pool for the partition.
	  ▪ **IE** - The Entry/Exit pool of the partition.
	  ▪ **drive** - A drive assigned to the partition.
	  ▪ **other** - Some other storage element.
	▪ **elementOffset** - The BlueScale number assigned to the slot or drive in the specified pool. This number appears as the number for each slot or drive in the BlueScale Inventory screen. Refer to your library *User Guide* for information about the Inventory screen.

firstRobot	Information about the first robot involved in the move. ▪ **number** - The number of the robot in the library. Numbering is from left to right as viewed from the front of the library. Values: **1** (left robot) or **2** (right robot). ▪ **lastOperation** - The type of operation last attempted by the robot. Values: **pick tape from slot**, **put tape to slot**, **pick magazine from drawer**, **put magazine to drawer**, **pick tape from drive**, **put tape to drive**, **other**
secondRobot	Information about the second robot involved in the move, if applicable. ▪ **number** - The number of the robot in the library. Numbering is from left to right as viewed from the front of the library. Values: **1** (left robot) or **2** (right robot). ▪ **lastOperation** - The type of operation last attempted by the robot. Values: **pick tape from slot**, **put tape to slot**, **pick magazine from drawer**, **put magazine to drawer**, **pick tape from drive**, **put tape to drive**, **other**

Example Command and Response The following command:

```
libraryStatus.xml?action=RCMStatus&id=FR1/RCM
```

immediately returns the following XML formatted response:

```
<libraryStatus>
   <moveOperationDetails>
      <move>
         <startTime>8/3/2016 13:25:56</startTime>
         <endTime>8/3/2016 13:27:41</endTime>
         <overallStatus>success</overallStatus>
         <overallSense>0x0, 0x00, 0x00</overallSense>
         <source>
            <partition>Partition 1</partition>
            <elementType>drive</elementType>
            <elementOffset>1</elementOffset>
         </source>
         <destination>
            <partition>Partition 1</partition>
            <elementType>storage</elementType>
            <elementOffset>36</elementOffset>
         </destination>
         <firstRobot>
            <number>2</number>
            <lastOperation>other</lastOperation>
         </firstRobot>
      </move>
      ...
      <move>
         <startTime>8/3/2016 12:22:45</startTime>
         <endTime>8/3/2016 12:24:10</endTime>
         <overallStatus>fail</overallStatus>
         <overallSense>0x5, 0x3b, 0x0d</overallSense>
         <source>
            <partition>Partition 1</partition>
            <elementType>storage</elementType>
            <elementOffset>1</elementOffset>
         </source>
         <destination>
            <partition>Partition 1</partition>
            <elementType>drive</elementType>
            <elementOffset>1</elementOffset>
         </destination>
         <firstRobot>
            <number>2</number>
            <lastOperation>put tape to drive</lastOperation>
         </firstRobot>
      </move>
   </moveOperationDetails>
</libraryStatus>
```

RCMStatus

Description Displays the status of the specified RCM.

Notes:
- This action is only supported on T950 and TFinity libraries.
- This action was added with BlueScale12.6.45.5.

Syntax `libraryStatus.xml?action=RCMStatus&id=[RCM ID]`

where the value for:

This parameter...	Specifies...
id	The component identifier for the RCM for which you want to retrieve status, using the form **FR[*integer*]/RCM**, where: ▪ **FR[*integer*]** = The designator for the frame. ▪ **RCM** = Indicates the RCM. **Notes:** ▪ If the library is not able to communicate with the RCM, the command returns a syntax error indicating an invalid **id**. ▪ A list of all RCMs the library can communicate with can be found in the ECInfo section of the response to the **libraryStatus.xml** command without any parameters. RCM component identifiers are preceded with "RCM Spectra PC" (see libraryStatus.xml [no parameters] or list on page 73).

Command Response The command immediately returns the following XML-formatted data:

```
<libraryStatus>
   <RCMStatus>
      <ID>formatted ID of RCM</ID>
      <overallStatus>healthy|poor</overallStatus>
      <loglibStatus>healthy|poor</loglibStatus>
      <motionStatus>healthy|poor</motionStatus>
      <repeaterStatus>healthy|poor</repeaterStatus>
   </RCMStatus>
</libraryStatus>
```

where the value for:

This parameter...	Indicates...
ID	The component identifier for the RCM using the form **FR[*integer*]/RCM**, where: ▪ **FR[*integer*]** = The designator for the frame. ▪ **RCM** = Indicates the RCM.
overallStatus	The status of the RCM as a whole. Values: **healthy**, **poor**
loglibStatus	The status of the LogLib application running on the RCM. Values: **healthy**, **poor**
motionStatus	The status of the motion application running on the RCM. Values: **healthy**, **poor** **Note:** This status is only returned for the main frame RCM.
repeaterStatus	The status of the repeater application running on the RCM. Values: **healthy**, **poor**

Example Command and Response The following command:

```
libraryStatus.xml?action=RCMStatus&id=FR1/RCM
```

immediately returns the following XML formatted response:

```
<libraryStatus>
   <RCMStatus>
      <ID>FR1/RCM</ID>
      <overallStatus>healthy</overallStatus>
      <loglibStatus>healthy</loglibStatus>
      <motionStatus>healthy</motionStatus>
      <repeaterStatus>healthy</repeaterStatus>
   </RCMStatus>
</libraryStatus>
```

refreshECInfo

Description Update the engineering change level information in the webserver cache with the current hardware settings.

Note: This action was added with BlueScale12.6.41.

Syntax

```
libraryStatus.xml?action=refreshECInfo[&extraInformation=[string]]
```

where the value for:

This parameter...	Specifies...
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command immediately returns the following XML-formatted data:

```
<libraryStatus>
   <message>[message text]</message>
   <status>OK</status>
</libraryStatus>
```

Progress Use `libraryStatus.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
libraryStatus.xml?action=refreshECInfo
```

immediately returns the following XML-formatted data:

```
<libraryStatus>
   <message>Started Refresh EC Info Action. Set progress in your
      query for status.</message>
   <status>OK</status>
</libraryStatus>
```

When the webserver cache update is complete, the response to `libraryStatus.xml?progress` contains `<status>OK</status>`.

refresh Environment

Description Update the environment information in the webserver cache with the current hardware settings.

Note: This action was added with BlueScale12.6.41.

Syntax `libraryStatus.xml?action=refreshEnvironment [&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command immediately returns the following XML-formatted data:

```
<libraryStatus>
   <message>[message text]</message>
   <status>OK</status>
</libraryStatus>
```

Progress Use `libraryStatus.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
libraryStatus.xml?action=refreshEnvironment
```

immediately returns the following XML-formatted data:

```
<libraryStatus>
   <message>Started Refresh Environment Action. Set progress in
      your query for status.</message>
   <status>OK</status>
</libraryStatus>
```

When the webserver cache update is complete, the response to `libraryStatus.xml?progress` contains `<status>OK</status>`.

CHAPTER 11

login

login.xml

Use the **login.xml** command to log a user into the library for the purpose of issuing additional XML commands to the library and checking the status of previously entered commands.

username

Description Connects to the library using the specified username and password. See "Configuring Library Users" in your library *User Guide* for information about configuring users and passwords, as well as information about what sort of actions each user type can perform.

Important The connection to the library is automatically closed after the idle time specified through the BlueScale user interface System Setup screen (see "Auto Logout Timeout" in your library *User Guide*) or can be closed by issuing a logout command, see logout.xml on page 100.

Important Connections to the library through the XML command interface are included in the maximum of eight simultaneous remote sessions supported by the library.

Syntax

`login.xml?username=[username]&password=[password]&forceFrontPanel`

where the value for:

This parameter...	Specifies...
username	A valid username assigned to the library. **Notes:** ▪ The specified user must have either superuser or administrator privileges in order to perform configuration operations. See "Understanding User Groups and Security" in your library *User Guide* for more information. ▪ Users assigned to the Operator group can move, import, and export media, but cannot access the more sensitive library operations such as configuration, diagnostics, and security. ▪ The username is case sensitive. ▪ To add a user, see "Add a New User" in your library *User Guide*.

This parameter...	Specifies...
password (optional)	The password associated with the username. **Notes:** ▪ If no password is set, this parameter is optional (`login.xml?username=su`) or can be blank (`login.xml?username=su&password=`). ▪ You may need to use HTML encoding to send special characters (%23, %40, etc.)
forceFrontPanel (optional)	Any future use of the BlueScale web interface (not the XML interface) using this created session will have access to all functions including those usually only available on the front panel.

Command Response The command immediately returns the following XML-formatted data:

```
<login>
   <status>OK</status>
</login>
```

The login command also returns a session ID cookie required for subsequent XML commands issued from the same location.

Important

If you are using a web browser to send commands, the cookie is handled transparently by the browser. If you are sending XML commands using a scripting language, the script must include commands to manage the session.

Example Command The following command:

```
login.xml?username=su&password=&forceFrontPanel
```

logs into the library as the default superuser, which, in this example, does not have a password set. When the BlueScale web interface is subsequently accessed using the same session cookie returned from this command, all front panel functionality is available.

CHAPTER 12

logout

logout.xml

Use the **logout.xml** command to close the connection to the library.

Important If you do not use the **logout.xml** command, the connection to the library is automatically closed after the idle time specified through the BlueScale user interface System Setup screen (see "Auto Logout Timeout" in your library *User Guide*).

Important Connections to the library through the XML command interface are included in the maximum of eight simultaneous remote sessions supported by the library.

Note: This command was added with BlueScale12.5.0.

Syntax `logout.xml`

Command Response The command immediately returns the following XML-formatted data:

```
<logout/>
```

Example Command The following command:

```
logout.xml
```

immediately closes the connection to the library. To run another XML command, you must first login using the **login.xml** command (see login.xml on page 98).

CHAPTER 13

mediaExchange

mediaExchange.xml

Use the **mediaExchange.xml** command to import, export, or exchange cartridges and magazines using the TeraPack® Access Port (TAP) or the bulk TAP, if present. The command also lets you prepare the bulk TAP for imports by ensuring that it does not contain any previously imported or exported magazines.

Note: The **mediaExchange.xml** command is not valid for the T120 library. Refer to the *Spectra T120 Library User Guide* for information about importing cartridges into or exporting cartridges from a T120.

Action	
clean	this page
getTAPState	page 103
importExport	page 106
Importing, Exporting, or Exchanging Cartridges	page 110
prepareImportExportList	page 113

clean

Description Prepares the specified bulk TAP for an import/export command by making sure that it is empty and that the TAP door is locked.

Notes:

- This action is only supported by libraries that have a bulk TAP.
- If the bulk TAP contains one or more magazines, the magazines are moved to the storage chambers assigned to the partition specified in the command. If the magazines belong to multiple partitions you need to use importExport on page 106 to import the magazines into the correct partitions.
- The clean action does not set the door release button LED to green nor does it open the bulk TAP door. Both the button and the door only operate during an import or export operation.

Syntax `mediaExchange.xml?action=clean&partition=[partition name]&TAPDevice=[main|leftBulk|rightBulk|leftAndRightBulk]&extraInformation=[string]`

where the value for:

This parameter...	Specifies...
partition	The exact name of the partition for which you want to clean the bulk TAP. Use the **partitionList.xml** command to retrieve a list of all the partitions currently configured in the library (see partitionList.xml on page 171). **Notes:** ▪ The partition name is case-sensitive. ▪ The partition name is set when the partition is created.
TAPDevice	The TAP for which cleaning is being requested. Values: ▪ **main** = The TAP in the main frame. ▪ **leftBulk** = The bulk TAP on the left end of the library. ▪ **rightBulk** = The bulk TAP on the right end of the library. ▪ **leftAndRightBulk** = The bulk TAPs on both the right and left end of the library. **Notes:** ▪ The **main** parameter value is not currently supported. The command always fails if it contains the **main** parameter value. ▪ The **leftBulk and rightBulk** parameter values are supported by T950 and TFinity libraries with a bulk TAP. The **leftAndRightBulk** parameter value is supported by TFinity libraries with a bulk TAP on each end of the library. The command fails if the library does not have a bulk TAP installed on the side specified by the parameter value.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<mediaExchange>
   <message>[message text]</message>
   <status>OK</status>
</mediaExchange>
```

Progress Use `mediaExchange.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
mediaExchange.xml?action=clean&parition=Partition 1&
   TAPDevice=leftBulk
```

immediately returns the following XML-formatted data:

```
<mediaExchange>
   <message>Started Bulk TAP clean.Set progress in your
      query for status.</message>
   <status>OK</status>
</mediaExchange>
```

The library checks the left bulk TAP to make sure it is empty and that the door is locked. If the bulk TAP contains one or more magazines, the magazines are moved to storage chambers in Partition 1. When the clean operation is complete, the response to `mediaExchange.xml?progress` contains `<status>OK</status>`.

getTAPState

Description Requests the status of the specified TAP.

Syntax

```
mediaExchange.xml?action=getTAPState&
   TAPDevice=[mainTop|mainBottom|leftBulk|rightBulk]&
   drawerNumber=[value]
```

where the value for:

This parameter...	Specifies...
TAPDevice	The TAP for which status is being requested. Values: ▪ **mainTop** = The top chamber of the TAP in the main frame. ▪ **mainBottom** = The bottom chamber of the TAP in the main frame. ▪ **leftBulk** = The bulk TAP on the left end of the library. ▪ **rightBulk** = The bulk TAP on the right end of the library. **Notes:** ▪ For T200 and T380 libraries, only the **mainTop** parameter value is supported. ▪ The **leftBulk and rightBulk** parameter values are supported by T950 and TFinity libraries with a bulk TAP. The **leftAndRightBulk** parameter value is supported by TFinity libraries with a bulk TAP on each end of the library. The command fails if the library does not have a bulk TAP installed on the side specified by the parameter value.
drawerNumber	**Optional.** The chamber (drawer) in the specified TAP for which status is being requested. Values: **1** through **14**, Default = **1** **Notes:** ▪ For the **mainTop** and **mainBottom** parameters, the value for **drawerNumber** is always **1**. ▪ The chambers in the bulk TAP carousel are numbered from right to left and top to bottom when viewed from the front of the library.

Command Response The command immediately returns the following XML-formatted data:

```
<mediaExchange>
   <doorOpen>true|false</doorOpen>
   <magazinePresent>true|false</magazinePresent>
   <magazineSeated>true|false</magazineSeated>
   <magazineType>
      LTO|LTO Maintenance|TS11x0|TS11x0 Maintenance|T10K|Unknown
   </magazineType>
   <rotaryPosition>
      UserSide|RobotSide|Unknown|Uninitialized
   </rotaryPosition>
</mediaExchange>
```

where the value for:

This parameter...	Indicates...
doorOpen	Whether the TAP door is open. Values: **true**, **false**
magazinePresent	Whether a magazine is present in the specified chamber. Values: **true**, **false** **Note:** For the **leftBulk** and **rightBulk** command parameters, the value for the **magazinePresent** parameter is only accurate when the value for the **rotaryPosition** parameter is **RobotSide**. That is, the bulk TAP carousel must be facing the interior of the library before the magazine states can be determined.
magazineSeated	Whether the magazine is properly seated in the chamber. Values: **true**, **false** **Notes:** ▪ The **magazineSeated** parameter is only returned if the value for the **magazinePresent** parameter is **true**. ▪ For the **leftBulk** and **rightBulk** command parameters, the value for the **magazineSeated** parameter is only accurate when the value for the **rotaryPosition** parameter is **RobotSide**. That is, the bulk TAP carousel must be facing the interior of the library before the magazine states can be determined.
magazineType	The magazine type. Values: **LTO**, **LTO Maintenance**, **TS11x0**, **TS11x0 Maintenance**, **T10K**, **Unknown** **Notes:** ▪ The **magazineType** parameter is only returned if the value for the **magazinePresent** parameter is **true** and the **TAPDevice** is **leftBulk** or **rightBulk**. ▪ The **TS11x0** and **TS11x0 Maintenance** values are only supported in T380, T950, and TFinity libraries that have TS11*x*0 technology drives installed. ▪ The **T10K** value is only supported in TFinity libraries that have T10K drives installed.

This parameter...	Indicates...
rotaryPosition	The position of the bulk TAP carousel. Values: ▪ **UserSide** = The bulk TAP carousel is facing the outside of the library. ▪ **RobotSide** = The bulk TAP carousel is facing the interior of the library. ▪ **Unknown** = The library is unable to determine the position of the carousel. ▪ **Uninitialized** = The bulk TAP has not completed its initialization process. **Note:** **rotaryPosition** is only shown when the **TAPDevice** is **leftBulk** or **rightBulk**.

Example Commands and Responses The following command:

```
mediaExchange.xml?action=getTAPState&TAPDevice=mainTop&
   drawerNumber=1
```

retrieves the status of the chamber in the top of the main frame TAP, indicating that the top TAP chamber contains an LTO magazine.

```
<mediaExchange>
   <doorOpen>false</doorOpen>
   <magazinePresent>true</magazinePresent>
   <magazineSeated>true</magazineSeated>
   <magazineType>LTO</magazineType>
   <rotaryPosition>Unknown</rotaryPosition>
</mediaExchange>
```

The following command:

```
mediaExchange.xml?action=getTAPState&TAP=leftBulk&drawerNumber=12
```

retrieves the status of chamber 12 in the left bulk TAP, indicating that the chamber contains an LTO magazine and that the bulk TAP carousel is rotated to face the interior of the library.

```
<mediaExchange>
   <doorOpen>false</doorOpen>
   <magazinePresent>true</magazinePresent>
   <magazineSeated>true</magazineSeated>
   <magazineType>LTO</magazineType>
   <rotaryPosition>RobotSide</rotaryPosition>
</mediaExchange>
```

importExport

Description Imports magazines into or exports them from the specified partition using the specified TAP.

Requirements When using this command, keep the following requirements in mind.

- The direction of the move is determined by whether the first location specified in the list of **TeraPackOffsets** is empty or full. If the first location is full, the moves performed by the command are export operations. If the first location is empty, the moves performed by the command are import operations.
- For a single command, all of the locations specified by the values for the **TeraPackOffsets** parameter must either be all empty or all full. A single command cannot mix imports and exports. Use the **physInventory.xml** command to determine the status of each inventory location in the partition (see physInventory.xml on page 172).

Important

The **offset** values given by **physInventory.xml** are one-based. The **TeraPackOffsets** required by **mediaExchange.xml** are zero-based. You must subtract 1 from the **offset** values before supplying them as **TeraPackOffsets**.

- No user intervention is possible while the command is being processed. Before running the command, issue the clean command, `mediaExchange.xml?action=clean&partition=[Partition name]`, for the bulk TAP you plan to use (see clean on page 101). The clean command ensures that the selected bulk TAP is empty and that the bulk TAP door is closed.

Important

If the carousel contains magazines from a previous export or import operation, the command fails. You must run the `mediaExchange.xml?action=clean` command to empty the bulk TAP before you attempt the next import or export operation (see clean on page 101).

- If you are exchanging magazines, remove each magazine from the carousel and replace it with another one of the same type. If you are exchanging or removing individual cartridges, remove the magazines from the carousel one at a time, make the desired cartridge changes, and then put the magazine back into the same location that it originally occupied.

Caution

When you place a magazine in the bulk TAP, make sure that the textured surface on each side of the magazine is toward the inside of the library and that the guides on the sides of the magazine fit into the media guides on the media shelf. Loading the magazines incorrectly or at an angle can result in damage to the carousel or the robotics.

Note: To ensure that you place the magazine in the same location that it originally occupied in the carousel, remove one magazine at a time.

- During an import using the bulk TAP, the transporter moves the magazines from the bulk TAP to the specified inventory locations in the selected partition. If you insert more magazines than the number of values for the **TeraPackOffsets** parameter, the extra magazines will be left in the carousel. You must then run the `mediaExchange.xml?clean` command for the partition and bulk TAP to remove the magazines before you can perform another import or export operation.

Syntax

```
mediaExchange.xml?action=importExport&
  partition=[partition name]&slotType=[storage|IE]&
  TAPdevice=[main|leftBulk|rightBulk|leftAndRightBulk]&
  timeoutInMinutes=[value]&TeraPackOffsets=[n,n,n...]
  [&extraInformation=[string]]
```

where the value for:

This parameter...	Specifies...
partition	The exact name of the partition into which you want to import or export the magazines. Use the **partitionList.xml** command to retrieve a list of all the partitions currently configured in the library (see partitionList.xml on page 171). **Notes:** ▪ The partition name is case-sensitive. ▪ The partition name is set when the partition is created.
slotType	The pool to be used for the import or export operation. Values: **storage**, **IE** (entry/exit) **Note:** Refer to "Media Pools" in your library *User Guide* for detailed information about the storage and entry/exit pools and their use.
TAPDevice	The TAP that will be used for the import or export operation. Values: ▪ **main** = The TAP in the main frame. ▪ **leftBulk** = The bulk TAP on the left end of the library. ▪ **rightBulk** = The bulk TAP on the right end of the library. ▪ **leftAndRightBulk** = The bulk TAPs on both the right and left end of the library. **Notes:** ▪ The **main** parameter value is not currently supported. The command always fails if it contains the **main** parameter value. ▪ The **leftBulk and rightBulk** parameter values are supported by T950 and TFinity libraries with a bulk TAP. The **leftAndRightBulk** parameter value is supported by TFinity libraries with a bulk TAP on each end of the library. The command fails if the library does not have a bulk TAP installed on the side specified by the parameter value.

This parameter...	Specifies...
timeoutInMinutes	The time, in minutes, that the library will wait for the door release button on the bulk TAP to be pushed, opening the bulk TAP door. Value: **1** through **10080** minutes (7 days), default = **10 minutes** **Important:** If you do not open the door within the number of minutes specified by the **timeoutInMinutes** parameter, the operation is aborted. The LED turns off and the carousel rotates to face the interior of the library. If magazines were left in the bulk TAP, you must run the `mediaExchange.xml?action=clean` command to empty the bulk TAP and then repeat the original command. **Note:** The **timeoutInMinutes** parameter only applies to the current command.
TeraPackOffsets	A comma-separated list of the offset values for the magazines to be moved. These offsets identify virtual locations in the partition inventory. Values: Use the `physInventory.xml?action=partition` command to obtain the offset value for the inventory locations you want to use (see physInventory.xml on page 172). **Important:** The **offset** values given by **physInventory.xml** are one-based. The **TeraPackOffsets** required by **mediaExchange.xml** are zero-based. You must subtract 1 from the **offset** values before supplying them as **TeraPackOffsets**. **Note:** The full or empty state of the first location in the list determines whether an import or export operation will be performed. The list must either contain all empty or all full locations.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<mediaExchange>
   <message>[message text]</message>
   </message>
   <status>OK</status>
</mediaExchange>
```

When the bulk TAP door release button illuminates solid green, press the button and insert, remove, or exchange the magazines (or the cartridges in the magazines) as required.

Progress The `mediaExchange.xml?action=importExport` operations are background operations. You can submit other commands while the background task is running. Use the following command to monitor the progress of the `importExport` command.

```
mediaExchange.xml?progress&TAPDevice=[main|leftBulk|rightBulk|
leftAndRightBulk]
```

where the value for:

This parameter...	Specifies...
TAPDevice	The TAP for which import or export operation is being requested. Values: ▪ **main** = The TAP in the main frame. ▪ **leftBulk** = The bulk TAP on the left end of the library. ▪ **rightBulk** = The bulk TAP on the right end of the library. ▪ **leftAndRightBulk** = The bulk TAPs on both the right and left end of the library. **Notes:** ▪ The **main** parameter value is not currently supported. The command always fails if it contains the **main** parameter value. ▪ The **leftBulk and rightBulk** parameter values are supported by T950 and TFinity libraries with a bulk TAP. The **leftAndRightBulk** parameter value is supported by TFinity libraries with a bulk TAP on each end of the library. The command fails if the library does not have a bulk TAP installed on the side specified by the parameter value.

The command returns the following XML-formatted data:

```
<progress>
   <activePage>[mediaexchange.xml]</activePage>
   <message>[running|notRunning]</message>
   <status>[Active|OK]</status>
   <extraInformation>[string]</extraInformation>
</progress>
```

where the value for:

This parameter...	Indicates...
activePage	Is the base URL. Value: **mediaExchange.xml**
message	The state of the import/export operation. Values: **running**, **notRunning**
status	Whether the magazine is properly seated in the chamber. Values: ▪ **Active** = the import/export/exchange operation is running. The corresponding message is **running**. ▪ **OK** = The import/export/exchange operation is complete. The corresponding message is **notRunning**.
extraInformation	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Example Command After using the **physInventory.xml** command to determine the offset values for empty chambers in a partition and the `mediaExchange.xml?action=clean` command to make sure that the bulk TAP is empty, use the following command to import magazines into the empty storage chambers in a partition named Partition 1:

```
mediaExchange.xml?action=importExport&partition=Partition 1&
   slotType=storage&TAPDevice=leftBulk&TeraPackOffsets=2,3,4,5
```

The library immediately returns the following XML-formatted data:

```
<mediaExchange>
   <message>
      Media export/import started for main|leftBulk|rightBulk TAP
      extraInformation=[string] (optional)
   </message>
   <status>OK</status>
</mediaExchange>
```

The bulk TAP carousel rotates to face the door. When the bulk TAP door release button illuminates green, press the button to open the door and insert the magazines. When you close the door, the robotics move the magazines to the specified chambers.

Note: Because the **timeoutInMinutes** parameter was not included in the command, the library uses the default timeout of 10 minutes. This means you have 10 minutes to press the button to open the bulk TAP door, insert the magazines in the carousel, and close the bulk TAP door.

IMPORTING, EXPORTING, OR EXCHANGING CARTRIDGES

The steps in the following examples illustrate the command sequences for using the bulk TAP to import or export one or more magazines.

Notes:
- You must be logged into the library using the **login.xml** command before issuing commands.
- Refer to the "Importing and Exporting Cartridges" chapter of your library *User Guide* for detailed information about preparing and using cartridges and magazines in the library and for detailed instructions for loading cartridges into the bulk TAP carousel.

Preparation

1. Run the following command to retrieve a list of all occupied magazines and cartridge locations in the partition named Partition 1. Use this data to determine the offset values for the magazines you want to use for your import, export, or exchange operation. See physInventory.xml on page 172 for details.

   ```
   physInventory.xml?partition=Partition 1
   ```

2. Run the following command to make sure that the left bulk TAP is empty and that the door is locked.

```
mediaExchange.xml?action=clean&partition=Partition 1&
   TAPDevice=leftBulk
```

3. Run the following command to determine when the clean action is complete.

```
mediaExchange.xml?progress
```

   Repeat the progress command as often as desired until the response is **OK**.

Import Magazines Example

1. After performing the steps in Preparation on page 110, run the following command to import six magazines into the entry/exit pool for Partition 1.

   Note: This example uses the data from the example command response for the `physInventory.xml?partition=Partition 1` command, beginning on page 175.

Important

The **offset** values given by **physInventory.xml** are one-based. The **TeraPackOffsets** required by **mediaExchange.xml** are zero-based. You must subtract 1 from the **offset** values before supplying them as **TeraPackOffsets**.

```
mediaExchange.xml?action=importExport&partition=Partition 1&
   slotType=IE&TAPDevice=leftBulk&
   TeraPackOffsets=60,61,62,63,64,65
```

2. From the front of the bulk TAP, wait for the bulk TAP door release button to illuminate solid green.
3. Press the bulk TAP door release button to open the bulk TAP door.
4. Insert six magazines into the chambers in the bulk TAP carousel.

Caution

When you place a magazine in the bulk TAP, make sure that the textured surface on each side of the magazine is toward the inside of the library and that the guides on the sides of the magazine fit into the media guides on the media shelf. Loading the magazines incorrectly or at an angle can result in damage to the carousel or the robotics.

   Note: Since no timeout was specified, you have 10 minutes to press the button to open the bulk TAP door, place the magazines in the carousel, and close the bulk TAP door.

5. After all of the magazines are loaded into the bulk TAP, close the bulk TAP door firmly. An audible click indicates that the door is latched closed. The carousel rotates to the interior of the library and the transporter begins moving the magazines to the chambers specified by the **TeraPackOffset** values.

6. Run the following command to determine when the import operation is complete.

   ```
   mediaExchange.xml?progress&TAPDevice=leftBulk
   ```

   Repeat the progress command as often as desired until the response contains `<status>OK</status>`.

Export Magazines Example

1. After performing the steps in Preparation on page 110, run the following command to export magazines from the storage pool for Partition 1.

   Note: This example uses the data from the example command response for the `physInventory.xml?partition=Partition 1` command, beginning on page 175.

Important The **offset** values given by **physInventory.xml** are one-based. The **TeraPackOffsets** required by **mediaExchange.xml** are zero-based. You must subtract 1 from the **offset** values before supplying them as **TeraPackOffsets**.

   ```
   mediaExchange.xml?action=importExport&partition=Partition 1&
      slotType=storage&TAPDevice=leftBulk&TeraPackOffsets=0,5
   ```

   The library retrieves the two magazines and places them in the left bulk TAP carousel.

2. When all of the magazines have been retrieved or when the bulk TAP is full, the carousel rotates to face the outside of the library and the door release button LED illuminates solid green.
3. From the front of the bulk TAP, press the bulk TAP door release button to open the bulk TAP door.
4. Remove the magazines from the bulk TAP.

   Note: Since no timeout was specified, you have 10 minutes to press the button to open the bulk TAP door, remove the magazines in the carousel, and close the bulk TAP door.

5. Close the bulk TAP door firmly. An audible click indicates that the door is latched closed. The carousel rotates to the interior of the library.
6. Run the following command to determine when the operation is complete.

   ```
   mediaExchange.xml?progress&TAPDevice=leftBulk
   ```

   Repeat the progress command as often as desired until the response contains `<status>OK</status>`.

prepare ImportExport List

Description This command creates a media exchange move file, listing the offsets for TeraPack magazine imports or exports. The move list must then be uploaded using the BlueScale Advanced Import/Export screen. See the *User Guide* for your library for more information.

Requirements For a single command, all of the locations specified by the values for the **TeraPackOffsets** parameter must either be all empty or all full. A single command cannot mix imports and exports. Use the **physInventory.xml** command to determine the status of each inventory location in the partition (see physInventory.xml on page 172).

Important The **offset** values given by **physInventory.xml** are one-based. The **TeraPackOffsets** required by **mediaExchange.xml** are zero-based. You must subtract 1 from the **offset** values before supplying them as **TeraPackOffsets**.

Syntax

```
mediaExchange.xml?action=prepareImportExportList&
    partition=[partition name]&slotType=[storage|IE]
    &TeraPackOffsets=[n,n,n...]
```

where the value for:

This parameter...	Specifies...
partition	The exact name of the partition into which you want to import or export the magazines. Use the **partitionList.xml** command to retrieve a list of all of the partitions currently configured in the library (see partitionList.xml on page 171). **Notes:** ▪ The partition name is case-sensitive. ▪ The partition name is set when the partition is created.
slotType	The pool to be used for the import or export operation. Values: **storage**, **IE** (entry/exit) **Note:** Refer to "Media Pools" in your library *User Guide* for detailed information about the storage and entry/exit pools and their use.
TeraPackOffsets	A comma-separated list of the offset values for the magazines to be moved. These offsets identify virtual locations in the partition inventory. Values: Use the `physInventory.xml?action=partition` command to obtain the offset value for the inventory locations you want to use (see physInventory.xml on page 172). **Important:** The **offset** values given by **physInventory.xml** are one-based. The **TeraPackOffsets** required by **mediaExchange.xml** are zero-based. You must subtract 1 from the **offset** values before supplying them as **TeraPackOffsets**. **Note:** The full or empty state of the first location in the list determines whether an import or export operation is performed. The list must either contain all empty or all full locations.

Command Response The command returns the following XML-formatted data:

```
<mediaExchange>
   <status>OK</status>
   <message>[message text]</message>
</mediaExchange>
```

Example Command After using the **physInventory.xml** command to determine the offset values for empty chambers in a partition, use the following command to create a move list for importing magazines into the empty storage chambers in a partition named Partition 1 using the BlueScale Advanced Import/Export screen:

```
mediaExchange.xml?action=prepareImportExportList&
   partition=Partition 1&slotType=storage&TeraPackOffsets=2,3,4,5
```

The library immediately returns the following XML-formatted data:

```
<mediaExchange>
   <status>OK</status>
   <message>Prepare Import Export List action successful</message>
</mediaExchange>
```

and creates a media exchange file on the library. When you navigate to the Advanced Import/Export screen in the BlueScale user interface, the **Populate** button displays. Click **Populate** to add the Imports in the media exchange file to the move queue. Click **Start Moves** to begin the imports.

CHAPTER 14

MLMSettings

MLMSettings.xml

Use the **MLMSettings.xml** command to list all Media Lifecycle Management (MLM) settings and enabled or disabled some settings. See your library *User Guide* for information about MLM settings.

Note: This command was added with BlueScale12.6.45.

Action	
list	this page
set	page 118

list

Description Returns a list of the current MLM settings.

Syntax `MLMSettings.xml?action=list`

Command Response The command immediately returns the following XML-formatted data:

```
<MLMSettings>
   <MLMEnabled>yes|no</MLMEnabled>
   <nonMLMAlertsEnabled>yes|no</nonMLMAlertsEnabled>
   <loadCountDiscrepancyAlertsEnabled>
      yes|no
   </loadCountDiscrepancyAlertsEnabled>
   <nonMLMLibraryLoadAlertsEnabled>
      yes|no
   </nonMLMLibraryLoadAlertsEnabled>
   <minCleaningPassesBeforeWarningCount>
      [integer]
   </minCleaningPassesBeforeWarningCount>
   <maxTapeLoadsBeforeWarningCount>
      [integer]
   </maxTapeLoadsBeforeWarningCount>
   <autoDiscoveryEnabled>yes|no</autoDiscoveryEnabled>
   <autoDiscoveryIdleWaitInMinutes>
      [integer]
   </autoDiscoveryIdleWaitInMinutes>
   <broadcastBaseConversion>
      none|DV25|DVCPRO25|MPEG_IMX30|MPEG_IMX40|MPEG_IMX50|DV50|
      DVCPRO-100|HDTV|DNX120|DDNX145|DNX185|DNX220|2K_film|
      DPX_film|4K_film|undefined
   </broadcastBaseConversion>
```

```
   <broadcastMegabitPerSecond>
      [integer]
   </broadcastMegabitPerSecond>
   <postScanTapeBlackout>
      <day of the week>
         <start>[integer]</start>
         <stop>[integer]</stop>
      </day of the week>
      ...
   </postScanTapeBlackout>
   <noncertifiedMAMBarcodeWriteEnabled>
      yes|no
   </noncertifiedMAMBarcodeWriteEnabled>
</MLMSettings>
```

where the value for:

This parameter...	Indicates...
MLMEnabled	Whether MLM is enabled (**yes**) or disabled (**no**). If MLM is enabled, Drive Lifecycle Management (DLM) is also enabled. Values: **yes**, **no**. Default: **yes**. **Note:** All other settings are ignored if <MLMEnabled> is no.
nonMLMAlerts Enabled	Whether MLM is configured to generate an alert message when a cartridge that is not Spectra MLM-enabled is loaded into a drive. Values: **yes**, **no**. Default: **no**.
loadCount DiscrepancyAlerts Enabled	Whether MLM is configured to generate an alert message when the load count for the cartridge stored in the MLM database differs from the load count stored on the cartridge's MAM (Medium Auxiliary Memory). Values: **yes**, **no**. Default: **no**.
nonMLMLibrary LoadAlertsEnabled	Whether MLM is configured to generate an alert message when the load count for the cartridge stored in the MLM database differs from the load count stored on the cartridge's MAM. Values: **yes**, **no**. Default: **no**.
minCleaningPasses BeforeWarning Count	The threshold for the minimum number of cleanings remaining on a cleaning cartridge. When an MLM-enabled cleaning cartridge reaches this threshold, a warning message is generated showing that the cleaning cartridge is nearly expired. The warning message is generated every time the cleaning tape is used while the number of cleanings remaining is at or below the threshold value. Default: **0**.
maxTapeLoads BeforeWarning Count	The number of times a tape can be loaded into a drive before a load count warning message is generated. When the number of tape loads reaches the specified threshold, a warning message is generated. Subsequent loads do not generate additional messages. Default: **10000**.
autoDiscovery Enabled	Whether the Media Auto Discovery feature, which discovers newly imported cartridges and adds them to the MLM database, is enabled. Values: **yes**, **no**. Default: **yes**.
autoDiscoveryIdle WaitInMinutes	The number of minutes that the library needs to be idle before the Media Auto Discovery process begins. Default: **2**.
broadcastBase Conversion	What broadcast base to use to convert capacity measurements into broadcast hours. Values: **none**, **DV25**, **DVCPRO25**, **MPEG_IMX30**, **MPEG_IMX40**, **MPEG_IMX50**, **DV50**, **DVCPRO-100**, **HDTV**, **DNX120**, **DDNX145**, **DNX185**, **DNX220**, **2K_film**, **DPX_film**, **4K_film**, **undefined**. Default: **none**.

This parameter...	Indicates...
broadcastMegabitPerSecond	The bit rate used for converting capacity to broadcast hours. The default depends on the broadcastBaseConversion chosen.
postScanTapeBlackout	The time periods during which the automatic PostScan process does not operate, specified in hours. Midnight is indicated by 0 (zero). For each day of the week (`sunday|monday|tuesday|wednesday|thursday|friday|saturday`), the following information is shown: ▪ **start** - The start time of the blackout period. ▪ **stop** - The end time of the blackout period. Default: no blackout period set (both start and stop set to 0 (zero)). **Note:** If both start and stop are set to 0 (zero), no blackout period is set for the day. If start is 0 (zero) and stop is 23, the blackout period is the full day.
noncertifiedMAMBarcodeWriteEnabled	Whether the barcode is written to the MAM for non-MLM-certified cartridges. Values: **yes**, **no**. Default: **no**.

Example Command and Response The following command:

```
MLMSettings.xml?action=list
```

immediately returns the following XML-formatted data:

```
<MLMSettings>
   <MLMEnabled>yes</MLMEnabled>
   <nonMLMAlertsEnabled>no</nonMLMAlertsEnabled>
   <loadCountDiscrepancyAlertsEnabled>
      no
   </loadCountDiscrepancyAlertsEnabled>
   <minCleaningPassesBeforeWarningCount>
      0
   </minCleaningPassesBeforeWarningCount>
   <maxTapeLoadsBeforeWarningCount>
      10000
   </maxTapeLoadsBeforeWarningCount>
   <autoDiscoveryEnabled>no</autoDiscoveryEnabled>
   <autoDiscoveryIdleWaitInMinutes>
      5
   </autoDiscoveryIdleWaitInMinutes>
   <broadcastBaseConversion>none</broadcastBaseConversion>
   <broadcastMegabitPerSecond>0</broadcastMegabitPerSecond>
```

```
   <postScanTapeBlackout>
      <sunday>
         <start>0</start>
         <stop>0</stop>
      </sunday>
      <monday>
         <start>0</start>
         <stop>0</stop>
      </monday>
      <tuesday>
         <start>0</start>
         <stop>0</stop>
      </tuesday>
      <wednesday>
         <start>0</start>
         <stop>0</stop>
      </wednesday>
      <thursday>
         <start>0</start>
         <stop>0</stop>
      </thursday>
      <friday>
         <start>0</start>
         <stop>0</stop>
      </friday>
      <saturday>
         <start>0</start>
         <stop>0</stop>
      </saturday>
   </postScanTapeBlackout>
   <noncertifiedMAMBarcodeWriteEnabled>
      no
   </noncertifiedMAMBarcodeWriteEnabled>
</MLMSettings>
```

set

Description Sets certain MLM options.

Notes:

- Disabling or enabling MLM may cause drives to reset. Do not proceed if there are operations currently running.
- Each MLMSettings command can only set one MLM setting. To change multiple settings, you must issue separate commands.

Syntax `MLMSettings.xml?action=set&MLM=[enable|disable] [&extraInformation=[string]]`

OR

`MLMSettings.xml?action=set&noncertifiedMAMBarcodeWrite=[enable|disable]`

where the value for:

This parameter...	Specifies...
MLM	Whether MLM is enabled. MLM is enabled by default. If MLM is enabled, Drive Lifecycle Management (DLM) is also enabled. Values: **enabled**, **disabled** **Note:** Media Lifecycle Management is not compatible with SlotIQ or soft load.
noncertifiedMAM BarcodeWrite	Whether the barcode for a non-MLM-certified cartridge is added to the cartridge's MAM. Values: **enabled**, **disabled**
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<MLMSettings>
   <status>OK</status>
   <message>[message text]</message>
</MLMSettings>
```

Progress If you enable or disable MLM, use `MLMSettings.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Note: Enabling or disabling `noncertifiedMAMBarcodeWrite` happens immediately. The progress command does not apply.

Example Command and Response The following command:

```
MLMSettings.xml?action=set&MLM=enable
```

immediately returns the following XML-formatted data:

```
<MLMSettings>
   <status>OK</status>
   <message>Started action to save MLM settings. Set progress
      in your query for status.</message>
   <message>MLM set to enable</message>
</MLMSettings>
```

When MLM is enabled, the response to `MLMSettings.xml?progress` contains `<status>OK</status>`.

CHAPTER 15

optionKeys

optionKeys.xml

Use the **optionKeys.xml** command to add a new option (activation) key to the library and to list all option keys currently active in the library. See "Enabling Bluescale Software Support, Options, And Upgrades" in your *User Guide* for more information about option keys.

Note: This command was added with BlueScale12.6.41.

Action	
add	this page
list	this page

add

Description Adds a new activation key to the library.

Note: Activation keys are tied to the library serial number (the hardware ID) and can only be used on the intended library.

Syntax `optionKeys.xml?action=add&key=[string]`

where the value for:

This parameter...	Indicates...
key	The alphanumeric key for activating the option. The key is not case-sensitive.

Command Response The command immediately returns the following XML-formatted data:

```
<optionKeys>
   <status>OK</status>
</optionKeys>
```

Example Command and Response The following command:

```
optionKeys.xml?action=add&key=GKM9VWS4C4HDDSC
```

immediately returns the following XML-formatted data and enables the appropriate option:

```
<optionKeys>
   <status>OK</status>
</optionKeys>
```

list

Description Returns a list of all active option keys currently entered in the library.

Syntax `optionKeys.xml?action=list`

Command Response The command immediately returns the following XML-formatted data:

```
<optionKeys>
   <optionKey>
      <keyValue>[key value]</keyValue>
      <description>[description]</description>
      <action>ADD|OR|OVERWRITE</action>
      <daysRemaining>[number of days remaining]</daysRemaining>
   </optionKey>
   ...
</optionKeys>
```

where the value for:

This parameter...	Indicates...
keyValue	The alphanumeric key for activating the option. The keys are not case-sensitive.
description	The option activated by the key.
action	Whether the key replaces, adds to, or coexists with previously entered keys of the same type. Values: ▪ **ADD -** The new quantity is added to previously entered keys of the same type. ▪ **OR**- Multiple keys of this type can coexist on the library. The feature associated with either the new key or previously entered keys of the same type can be used. ▪ **OVERWRITE** - The new key replaces previously entered keys of the same type. **EXAMPLE:** In the following example command response, the Capacity License key includes the total licensed capacity rather than incremental capacity and **Overwrites** previously entered Capacity License keys.
daysRemaining	The number of days remaining before the key expires, if applicable.

Example Command and Response The following command:

```
optionKeys.xml?action=list
```

immediately returns the following XML-formatted data:

```
<optionkeys>
   <optionKey>
      <keyValue>GKM9VWS4C4HDDSC</keyValue>
      <description>BlueScale Software Support License: Package
         Update Enabled</description>
      <action>OVERWRITE</action>
      <daysRemaining>310</daysRemaining>
   </optionKey>
   ...
   <optionKey>
      <keyValue>H86JVXE5QGWLGPD</keyValue>
      <description>Capacity License: 1000 Chambers</description>
      <action>OVERWRITE</action>
   </optionKey>
</optionkeys>
```

CHAPTER 16

package

package.xml

Use the **package.xml** command to do the following operations:

- Retrieve the name of the BlueScale package currently used by the library, as well as a list of all of the BlueScale package files currently stored on the memory card in the LCM
- Update the library using a BlueScale package previously uploaded to the memory card in the LCM

  Note: Use the **packageUpload.xml** command (see packageUpload.xml on page 139) to upload a BlueScale package to the memory card in the LCM.
- Check the results of an update operation
- See if all hardware components are at the correct firmware level for the currently installed BlueScale package

Action	
[no parameters] or list	this page
displayCurrentFirmwareVersions	page 125
displayPackageDetails	page 127
getResults	page 129
stagePackage	page 132
update	page 133
Updating the Library BlueScale Software	page 137

[no parameters] or list

Description Retrieves the name of the BlueScale package currently used by the library. The data returned by the command also lists all of the BlueScale package files currently stored on the memory card in the LCM.

Syntax `package.xml?action=list`

Command Response The command immediately returns the following XML-formatted data:

```
<package>
   <current>
      <name>[version name]</name>
   </current>
   <list>
      <name>[version name]</name>
      ...
      <name>[version name]</name>
   </list>
</package>
```

where the value for:

This parameter...	Indicates...
current	The container for the BlueScale software version currently running on the library. **Note:** If the library is not at a specific package level, then the value of the **name** parameter in the **current** section of the command response is **None**.
name	The package name of the BlueScale software version. The package name format is: **BlueScale[*n.n.n*]-[*YYYYMMDD*][*release type*]** where: ▪ ***n.n.n*** = The version number. ▪ ***YYYYMMDD*** = The date of the BlueScale package. ▪ ***release type*** = Either an **F** for full release or an **I** for incremental release.
list	The container for the list of BlueScale packages currently stored on the library. **Note:** If there are no package files stored on the library, the **list** section is not returned.

Example Command and Response The following command:

```
package.xml
```

returns the current BlueScale package information:

```
<package>
   <current>
      <name>BlueScale12.6.44.4-20150601F</name>
   </current>
   <list>
      <name>BlueScale12.6.44.4-20150601F</name>
      <name>BlueScale12.0.3-20111122F</name>
      <name>BlueScale11.3.3-20110316F</name>
   </list>
</package>
```

display Current Firmware Versions

Description Display the current firmware version installed on individual components in the library along with the firmware version included in the currently installed BlueScale package version.

Note: This action was added with BlueScale12.6.45.

Syntax `package.xml?action=displayCurrentFirmwareVersions`

Command Response The command immediately returns the following XML-formatted data:

```
<package>
   <packageName>[version name]</packageName>
   <allComponentsUpToDate>yes|no</allComponentsUpToDate>
   <component>
      <name>[component ID]</name>
      <currentVersion>[current version]</currentVersion>
      <packageVersion>[package version]</packageVersion>
   </component>
   ...
</package>
```

where the value for:

This parameter...	Indicates...
packageName	The full name of the BlueScale software version currently running on the library. The package name format is: **BlueScale[*n.n.n*]-[*YYYYMMDD*][*release type*]** where: ▪ ***n.n.n*** = The version number. ▪ ***YYYYMMDD*** = The date of the BlueScale package. ▪ ***release type*** = Either an **F** for Full release or an **I** for Incremental release. **Note:** If the library is not at a specific package level, then the value of the name parameter in the current section of the command response is **None**.
allComponentsUpToDate	Whether all components are at the firmware versions matching the BlueScale package indicated by **packageName**. Values: **yes, no**
component	The container for information about one component.
name	The description or component identifier for the component. **Note:** See the "Architecture" chapter in your library *User Guide* for detailed information about component identifiers.
currentVersion	The current firmware version running on the component.
packageVersion	The firmware version for the component in the BlueScale package indicated by **packageName**.

Example Command and Response The following command:

```
package.xml?action=displayCurrentFirmwareVersions
```

returns the current BlueScale package and library component version information:

```
<package>
   <packageName>BlueScale12.6.45-20150428F</packageName>
   <allComponentsUpToDate>yes</allComponentsUpToDate>
   <component>
      <name>LC OS</name>
      <currentVersion>100.7.44.4</currentVersion>
      <packageVersion>100.7.44.4</packageVersion>
   </component>
   <component>
      <name>LC Server</name>
      <currentVersion>4.15.45.1</currentVersion>
      <packageVersion>4.15.45.1</packageVersion>
   </component>
   <component>
      <name>FR1/PCM</name>
      <currentVersion>4.15.41.1</currentVersion>
      <packageVersion>4.15.41.1</packageVersion>
   </component>
   <component>
      <name>FR1/FMM</name>
      <currentVersion>4.15.41.1</currentVersion>
      <packageVersion>4.15.41.1</packageVersion>
   </component>
   <component>
      <name>FR1/DBA4/fLTO-DRV1</name>
      <currentVersion>4.15.45.1</currentVersion>
      <packageVersion>4.15.45.1</packageVersion>
   </component>
   <component>
      <name>FR1/DBA4/fLTO-DRV4</name>
      <currentVersion>4.15.45.1</currentVersion>
      <packageVersion>4.15.45.1</packageVersion>
   </component>
   <component>
      <name>FR1/DBA2/F-QIP1</name>
      <currentVersion>8.15.45.1</currentVersion>
      <packageVersion>8.15.45.1</packageVersion>
   </component>
   <component>
      <name>FR1/RCM</name>
      <currentVersion>4.15.45.2</currentVersion>
      <packageVersion>4.15.45.2</packageVersion>
   </component>
</package>
```

display Package Details

Description Display the current firmware version installed on individual components in the library along with the firmware version included in the specified BlueScale package version.

Note: This action was added with BlueScale12.7.02.

Syntax `package.xml?action=displayPackageDetails&package=[package name]`

where the value for:

This parameter...	Specifies...
package	The name of the BlueScale package to which you want to compare the currently installed firmware. The package name format is: **BlueScale[*n.n.n*]-[*YYYYMMDD*][*release type*]** where: ▪ ***n.n.n*** = The version number. ▪ ***YYYYMMDD*** = The date of the BlueScale package. ▪ ***release type*** = Either an **F** for full release or an **I** for incremental release.

Command Response The command immediately returns the following XML-formatted data:

```
<package>
   <packageName>[version name]</packageName>
   <allComponentsUpToDate>yes|no</allComponentsUpToDate>
   <allComponentsFullyStaged>yes|no</allComponentsFullyStaged>
   <component>
      <name>[component ID]</name>
      <currentVersion>[current version]</currentVersion>
      <packageVersion>[package version]</packageVersion>
      <fullyStaged>yes|no</fullyStaged>
   </component>
   ...
</package>
```

where the value for:

This parameter...	Indicates...
packageName	The full name of the BlueScale software version currently running on the library. The package name format is: **BlueScale[*n.n.n*]-[*YYYYMMDD*][*release type*]** where: ▪ ***n.n.n*** = The version number. ▪ ***YYYYMMDD*** = The date of the BlueScale package. ▪ ***release type*** = Either an **F** for Full release or an **I** for Incremental release. **Note:** If the library is not at a specific package level, then the value of the name parameter in the current section of the command response is **None**.
allComponentsUpToDate	Whether all components are at the firmware versions matching the BlueScale package indicated by **packageName**. Values: **yes, no**
allComponentsFullyStaged	Whether all components which allow firmware staging have firmware staging complete. Values: **yes, no**

This parameter...	Indicates...
component	The container for information about one component.
name	The description or component identifier for the component. **Note:** See the "Architecture" chapter in your library *User Guide* for detailed information about component identifiers.
currentVersion	The current firmware version running on the component.
packageVersion	The firmware version for the component in the BlueScale package indicated by **packageName**.
fullyStaged	Whether the new firmware is fully staged to the component and ready for the update. This only displays for components that allow staging: RCM, RIM, RIM2, and DCM. Values: **yes, no**

Example Command and Response The following command:

```
package.xml?action=displayPackageDetails&package=BlueScale12.7.02-
20170405F
```

returns the current BlueScale package and library component version information:

```
<package>
   <packageName>BlueScale12.7.02-20170405F</packageName>
   <allComponentsUpToDate>no</allComponentsUpToDate>
   <allComponentsFullyStaged>no</allComponentsFullyStaged>
   <component>
      <name>LC OS</name>
      <currentVersion>107.2.0.1</currentVersion>
      <packageVersion>107.2.0.3</packageVersion>
   </component>
   <component>
      <name>LC Server</name>
      <currentVersion>7.2.0.15</currentVersion>
      <packageVersion>7.2.0.23</packageVersion>
   </component>
   <component>
      <name>FR1/PCM</name>
      <currentVersion>7.2.0.1</currentVersion>
      <packageVersion>7.2.0.2</packageVersion>
   </component>
   <component>
      <name>FR1/FMM</name>
      <currentVersion>7.2.0.1</currentVersion>
      <packageVersion>7.2.0.1</packageVersion>
   </component>
   <component>
      <name>FR1/DBA4/fLTO-DRV1</name>
      <currentVersion>7.2.0.9</currentVersion>
      <packageVersion>7.2.0.13</packageVersion>
      <fullyStaged>no</fullyStaged>
   </component>
```

```
    <component>
       <name>FR1/DBA2/F-QIP1</name>
       <currentVersion>9.2.0.3</currentVersion>
       <packageVersion>9.2.0.4</packageVersion>
       <fullyStaged>no</fullyStaged>
    </component>
    <component>
       <name>FR1/RCM</name>
       <currentVersion>7.2.0.23</currentVersion>
       <packageVersion>7.2.0.36</packageVersion>
       <fullyStaged>no</fullyStaged>
    </component>
</package>
```

getResults

Description Returns status information for the most recent BlueScale package update, including all component firmware versions. If the package included updates to the LCM or RCM firmware, the command also reboots the LCM and RCM.

Important

Updates do not take effect until the library completes the update process and, if necessary, reboots the LCM and any other components that were updated. If you did not include the **autoFinish** parameter in the `package.xml?action=update` command (see update on page 133), you must send a `package.xml?action=getResults` command in order to complete the update process.

Notes:

- Running this action is not required in order to complete the update process if you include the **autoFinish** parameter in the `package.xml?action=update` command as described in autoFinish (optional) on page 135.
- The remote connection to the library is lost when the LCM reboots. Allow sufficient time for the LCM to complete its initialization, then reconnect to the library.

Syntax `package.xml?action=getResults`

Command Response If the last package update was done using `package.xml?action=update`, the command immediately returns the following XML-formatted data:

```
<package>
   <updateResults>
      <packageName>[package name]</packageName>
      <component>
         <name>[firmware component name]</name>
         <previousVersion>
            [previous firmware version]
         </previousVersion>
         <updatedVersion>
            [updated firmware version]
         </updatedVersion>
         <updateStatus>
            OK|failure message [message text]
         </updateStatus>
      </component>
      ...
      <rebootInProcess>true|false</rebootInProcess>
   </updateResults>
</package>
```

where the value for:

This parameter...	Indicates...
packageName	The name of the BlueScale package used to update the library in the format **BlueScale[*n.n.n*]-[*YYYYMMDD*][*release type*]** where: ▪ ***n.n.n*** = The version number. ▪ ***YYYYMMDD*** = The date of the BlueScale package. ▪ ***release type*** = Either an **F** for Full release or an **I** for Incremental release.
name	The name of the BlueScale component updated.
previousVersion	The version number of the BlueScale component before the update.
updatedVersion	The version number of the BlueScale component following the update.
updateStatus	The status of the update. Values: ▪ **OK** = The update completed successfully. ▪ **failure message** = An error message indicating why the update did not complete successfully.
rebootInProcess	Whether the library requires a reboot of the LCM and RCM to complete the update. Values: ▪ **true** = The library will reboot components to complete the update. ▪ **false** = No components were updated that require a reboot.

If the last update was done using a method other than `package.xml?action=update`, the command returns the following XML-formatted data:

```
<error>
   <message>
      Package update action failed or never started
   </message>
</error>
```

Example Command and Response The following command:

```
package.xml?action=getResults
```

returns the following information for the example shown on page 136:

```
<package>
   <updateResults>
      <packageName>BlueScale12.1.0-20120423F</packageName>
      <component>
         <name>LC OS</name>
         <previousVersion>100.4.0.0</previousVersion>
         <updatedVersion>100.5.0.0</updatedVersion>
         <updateStatus>OK</updateStatus>
      </component>
      <component>
         <name>LC Server</name>
         <previousVersion>4.7.0.27</previousVersion>
         <updatedVersion>4.8.0.59</updatedVersion>
         <updateStatus>OK</updateStatus>
      </component>
      <component>
         <name>LC Web Server</name>
         <previousVersion>4.7.0.27</previousVersion>
         <updatedVersion>4.8.0.59</updatedVersion>
         <updateStatus>OK</updateStatus>
      </component>
      ...
      <rebootInProcess>false</rebootInProcess>
   </updateResults>
</package>
```

stagePackage

Description Starts the background staging of firmware updates to all library components that support staging (RCM, RIM, RIM2, DCM).

Note: This action was added with BlueScale12.7.02.

Important The specified BlueScale package must already be stored on the library memory card. If the desired package is not present, you must first upload the necessary BlueScale package (see packageUpload.xml on page 139).

Important If you receive an error message stating that your disk is full when the library attempts to unzip an update package, see "Manage Update Packages" in your library *User Guide* for information about deleting packages from the memory card.

Syntax `package.xml?action=stagePackage&package=[package name]`

where the value for:

This parameter...	Specifies...
package	The name of the BlueScale package you want to use to update the library. The package name format is: **BlueScale[*n.n.n*]-[*YYYYMMDD*][*release type*]** where: ▪ ***n.n.n*** = The version number. ▪ ***YYYYMMDD*** = The date of the BlueScale package. ▪ ***release type*** = Either an **F** for full release or an **I** for incremental release.

Command Response The command returns the following XML-formatted data:

```
<package>
   <status>OK</status>
   <message>[message text]</message>
</package>
```

Example Command and Response The following command:

```
package.xml?action=stagePackage&package=BlueScale12.7.02-20170405F
```

immediately returns the following XML-formatted data:

```
<package>
   <status>OK</status>
   <message>
      Started firmware staging. Use action=displayPackageDetails
      for staging status.
   </message>
</package>
```

Run `displayPackageDetails.xml` to determine when all firmware is staged. See displayPackageDetails on page 127.

update **Description** Updates the library to the specified BlueScale package.

Important The specified BlueScale package must already be stored on the library memory card. If the desired package is not present, you must first upload the necessary BlueScale package (see packageUpload.xml on page 139).

Important If you receive an error message stating that your disk is full when the library attempts to unzip an update package, see "Manage Update Packages" in your library *User Guide* for information about deleting packages from the memory card.

Preparation Before you begin the package update process, make sure that you review and address the following requirements:

- Make sure your BlueScale Software Support key is current. Updating the BlueScale software and the firmware for library components requires a current service contract with Spectra Logic Technical Support. The update fails if the library does not have a valid BlueScale Software Support key installed.

  If your service contract expired, renew it as described in "Renewing the BlueScale Software Support Key" in your library *User Guide* for instructions.

- Stop all library operations.

  Notes:

  - If you are updating a library running BlueScale 12.7.00.05 or 12.7.00.06, at the start of an update that includes LCM, RCM, RIM, or RIM2 updates, the library shuts down all exporters (RIM and RIM2) so that they do not accept additional moves, and then waits for any moves in progress to complete before starting to update the selected components. The exporters are re-enabled to accept moves when the package update completes.
  - If you are updating a library running BlueScale 12.7.00.7 or later, at the start of a package update that includes RCM or LCM updates, the library shuts down all exporters (RIM, RIM2, and drive) so that they do not accept additional moves and then waits for any moves in progress to complete before starting to update the selected components. At the start of a package update that includes RIM or RIM2 updates only, the library shuts down the exporters needing an update so that they do not accept additional moves and then waits for any moves in progress to complete before starting to update the selected components. The exporters are re-enabled to accept moves when the package update completes.

Important Confirm that all of the following conditions are met before beginning the upgrade process:

- All host processes have completed.
- All storage management software daemons are stopped.

- Use your storage management software to move any cartridges that are currently in drives back to their storage locations. If you cannot use your storage management software, move the cartridges as described in your library *User Guide*.
- Pause or stop library background operations, if they are running. Any tapes currently being scanned are returned to their storage locations.
  a. Select **Maintenance** ···⁘ **MLM** to display the Media Lifecycle Management Tools screen.
  b. Click **Pause PostScan** to pause the PostScan operation for one hour.
  c. Click **Stop Discovery** to stop the PreScan operation.
- When updating a library running a version of BlueScale software earlier than BlueScale12.7.02, to ensure that no commands are sent to the controllers, Spectra Logic recommends disconnecting Fibre Channel cables connected to the F-QIPs, RIMs, and any tape drives used as robotic exporters.
- Back up your MLM database (see "Back Up the MLM and DLM Databases" in your library *User Guide* for instructions).

  Note: Backing up the MLM database also backs up the DLM database.
- Back up the library configuration (see "Backing Up the Library Configuration" in your library *User Guide* for instructions).
- Back up all of your BlueScale encryption keys (see "Exporting and Protecting Keys" in the *Spectra Encryption User Guide* for instructions).

Syntax `package.xml?action=update&package=[package name]&autoFinish [&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
package	The name of the BlueScale package to which you want to update the library. The package name format is: **BlueScale[*n.n.n*]-[*YYYYMMDD*][*release type*]** where: ▪ ***n.n.n*** = The version number. ▪ ***YYYYMMDD*** = The date of the BlueScale package. ▪ ***release type*** = Either an **F** for full release or an **I** for incremental release.
autoFinish (optional)	When included, the library automatically completes the update instead of waiting to receive an `update.xml?getResults` command. If the update package included updates to the LCM or RCM firmware, the library reboots the LCM and RCM when the update completes. The remote connection to the library is lost when the LCM reboots. Allow sufficient time for the LCM to complete its initialization, then reconnect to the library. **Important:** If the package includes updates to the LCM or RCM firmware and you do not include the **autoFinish** parameter, you must run the `update.xml?getResults` command to reboot the LCM and RCM to complete the update. See getResults on page 129.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<package>
   <status>OK</status>
   <message>[message text]</message>
</package>
```

Important The external lights on TFinity libraries change to yellow during the update process.

Important Once the update process starts, it cannot be canceled. Do not power off the library or any component during the update process.

Progress Use `package.xml?progress` to determine the status of the operation (see Progress for Extended Action Commands on page 19). If the command included the autoFinish parameter and the update included changes to the LCM or RCM, the updated component reboots as soon as the command completes. Otherwise, when the update completes, `<status>` in the command response is **OK** or **FAILED**.

Retrieve the results and complete the update operation using one of the following methods:

- If the command did not include the **autoFinish** parameter, issue an `package.xml?action=getResults` command to retrieve the results for the update and, if necessary, reboot the library to complete the update (see getResults on page 129 for more information).

Important

Updates do not take effect until the library completes the update process and, if necessary, reboots the LCM and any other components that were updated. If you did not include the **autoFinish** parameter in the command, you must send a `package.xml?action=getResults` command in order to complete the update process. See getResults on page 129 for more information.

- If the command included the **autoFinish** parameter but the update did not require a reboot, you can retrieve the update results using the **systemMessages.xml** command (see systemMessages.xml on page 186).
- If the command included the **autoFinish** parameter and the update required a reboot, you can retrieve the update results using the `traces.xml?traceType=Message` command (see traces.xml on page 192).

Example Command and Response The following command:

```
package.xml?action=update&package=BlueScale12.6.44.4-20150601F&
   autoFinish
```

immediately returns the following XML-formatted data:

```
<package>
   <status>OK</status>
   <message>
      Started firmware update. Set progress in your query for status.
   </message>
</package>
```

and updates the library to BlueScale12.6.44.4-20150601F. When the response to `libraryStatus.xml?progress` contains `<status>OK</status>`, the package update is complete. The LCM or RCM reboots automatically if the package included updates to the LCM or RCM firmware.

UPDATING THE LIBRARY BLUESCALE SOFTWARE

The following steps illustrate the sequence of commands used when updating the library to a new version of the BlueScale software.

Notes:
- You must be logged into the library using the **login.xml** command before you can issue any additional commands.
- Refer to "Updating the BlueScale Software and Library Firmware" in your library *User Guide* for detailed information about BlueScale update packages.

1. Run the program you created to transfer the BlueScale package to the library using the **packageUpload.xml** command (see packageUpload.xml on page 139).
2. Run the following command to confirm that the BlueScale package you uploaded is on the library memory card.

   ```
   package.xml
   ```

   The package you uploaded is listed in the **list** section of the XML data returned in the command response.
3. Run the following command to update the library to use the BlueScale package you uploaded and automatically reboot the LCM if the package included updates to the LCM or RCM firmware.

   ```
   package.xml?action=update&package=[package name]&autoFinish
   ```
4. Run the following command to check the progress of the update:

   ```
   package.xml?progress
   ```
5. When you receive an **OK** response, complete the update.
   - If you included the **autoFinish** parameter in Step 3, the library automatically completes the update and posts the results as a system message.
   - If you did not include the **autoFinish** parameter in Step 3, run the following command to complete the update and retrieve the results.

     ```
     package.xml?action=getResults
     ```

   Note: The remote connection to the library is lost when the LCM reboots. Allow sufficient time for the LCM to complete its initialization, then reconnect to the library.
6. Run one of the following commands to retrieve any system messages posted during the update process (see systemMessages.xml on page 186 and traces.xml on page 192 for descriptions of the commands).
   - If the update did not require a reboot of the LCM, use:
     ```
     systemMessages.xml
     ```
   - If the update required a reboot of the LCM, use:
     ```
     traces.xml?traceType=Message
     ```

7. Run the following command to confirm that the library is using the BlueScale package you uploaded.

   ```
   package.xml
   ```

   The package you used for the update should be listed in the **current** section of the XML data returned in the command response.

CHAPTER 17

packageUpload

packageUpload.xml

Use the **packageUpload.xml** command within a script to upload a BlueScale package file to the library's memory card. After uploading the package file, use the package to update the BlueScale software that the library is running (see package.xml on page 123).

Important The **packageUpload.xml** command uses an HTTP POST action to transfer the package file to the library. This cannot be accomplished through a browser. You must use a script to preform the upload.

Important If you are updating a library running BlueScale12.7.02 or later, upload a package ending with the letter "s", which indicates a digitally signed package. If you are updating a library running a BlueScale version prior to 12.7.02, upload a package ending with the letter "z", which indicates an unsigned zip package.

[no parameters]

Uploads the specified BlueScale package file to the library using the Ethernet connection to the LCM.

Syntax `packageUpload.xml`

Requirement Use a standard programming language such as Java, Perl, or Python to create the necessary program to transfer the BlueScale package to the library using the **packageUpload.xml** command.

Command Example and Response The following is an example Python 3.7 script for uploading a BlueScale package. This is compatible with libraries using a version of BlueScale software with TLS1.2. The following are the versions of BlueScale software that implemented TLS1.2 for each library type:

- T120: BlueScale12.7.07.03
- T200, T380, T680: not yet released
- T950: BlueScale12.7.06.01
- TFinity: BlueScale12.8.00

```
import argparse
import os
import urllib3
import requests
```

```
def make_address(library, ssl, element, args):
   if ssl:
      address = 'https://'
   else:
      address = 'http://'
   address += library + '/gf/' + element + '.xml'
   if args is not None:
      num_args = len(args)
      i = 0
      address += '?'
      for key in args:
         if isinstance(args[key], str):
            value = args[key].replace(" ", "%20")
            address += key + '=' + value
         elif isinstance(args[key], list):
            address += key + '=' + ','.join(args[key])
         elif args[key] is None:
            address += key
         i += 1
         if i != num_args:
            address += '&'
   return address

def login(session, library, ssl, user, password):
   address = make_address(library, ssl, 'login', {'username':
      user, 'password': password})
   CONNECTION_TIMEOUT_SEC = 10
   READ_TIMEOUT_SEC = None
   return session.get(address, verify=False,
      timeout=(CONNECTION_TIMEOUT_SEC, READ_TIMEOUT_SEC))

def upload_package(session, library, ssl, file_name):
   address = make_address(library, ssl, 'packageUpload', None)
   try:
      with open(file_name, "rb") as f:
         data = {"BlueScalePkg": f}
         response = session.post(address, files=data)
         if response.text.lower().count('ok') < 1:
            return {}, Error("Package file upload failed: " +
               response.text)
   except FileNotFoundError:
      return {}, Error("File " + str(file_name) + " not found.")
   except OSError as e:
      return {}, Error("Error " + str(e.errno) + " opening package
         file.")
   except BaseException:
      return {}, Error("A network error occurred while attempting
         to upload the package file.")
   return response, None

def main():
   parser = argparse.ArgumentParser()
   parser.add_argument('-f', '--file', help="package file")
   parser.add_argument('-l', '--library', help="library to
      update")
   parser.add_argument('-s', '--ssl', help="use ssl")
   args = parser.parse_args()
```

```
    if not args.file:
       args.file = input("Which package file should I use? ")

    if not os.path.isfile(args.file):
       print("Package file '%s' does not exist; exiting" %
          args.file)
       exit(2)

    if not args.library:
       args.library = input("Which library do you want to upload
          to? ")

    if not args.ssl:
       args.ssl = input("Should I use ssl (yes, no)? ")
    ssl = args.ssl in ['y', 'yes']

    urllib3.disable_warnings()
    session = requests.Session()
    response = login(session, args.library, ssl, 'su', '')
    if response.text.lower().count('ok') < 1:
       print("Login failed: %s" % response.text)
       exit(2)

    file_name = os.path.basename(args.file)
    pkg_name = os.path.splitext(file_name)[0]
    print("Uploading package %s to library %s" % (pkg_name,
       args.library))

    result, err = upload_package(session, args.library, ssl,
       args.file)
    if err is not None:
       print("Error: %s" % err.message)
    else:
       print("Successfully uploaded package %s to library %s" %
          (pkg_name, args.library))
    exit(0)

if __name__ == '__main__':
    main()
```

The command returns the following XML-formatted data:

```
<packageUpload>
   <status>OK</status>
</packageUpload>
```

CHAPTER 18

partition

partition.xml

Use **partition.xml** to configure one or more data or cleaning partitions in the library or to get information about previously configured partitions. See the "Configuring and Managing Partitions" chapter in your library *User Guide* for detailed information about configuring data and cleaning partitions.

Note: Use the **partitionList.xml** command to retrieve a list of the partitions currently configured on the library (see partitionList.xml on page 171). Use `partition.xml?list` to get a list of partitions with detailed information about the partition configuration (see list on page 147).

autoCreate

Description Automatically creates a single storage partition that uses all installed drives, has a single chamber assigned to the entry/exit pool (excluding T120), and has all or most of the remaining chambers (or slots in the T120 library) that are licensed by the CoD key assigned to the storage pool. In T200 and larger libraries , if the entire library is licensed, one chamber per frame is left out of the partition to optimize tape movements.

Requirements and Guidelines When preparing to use this command, keep the following requirements and guidelines in mind.

- This command can only be used if the library does not currently have any partitions configured. If there are already partitions configured on the library, the command fails.

- All of the drives and controllers (RIMs or F-QIPs) installed in the library must be of the same technology and use the same interface. For example, if one drive is a Fibre Channel LTO drive, then all of the drives must be Fibre Channel LTO drives.
- With the exception of the T120 library, one chamber in the library is reserved for the partition's entry/exit pool. As a result, the number of chambers assigned to the storage pool is one fewer than the total number of chambers enabled by the CoD key.
- For the T120 library, the eject mode is set to Standard.
- If you want to use Auto Drive Clean with the partition and the library does not contain any chambers that are not enabled by the CoD activation key, modify the partition after it is created to remove some of the chambers assigned to it so that you can create a cleaning partition.
- If you want to use the Global Spares option with the partition, modify the partition after it is created to unassign one or more of the drives that were automatically assigned to the partition and then reassign those drives as Global Spares.
- You may want to modify the partition after it is created to remove some of the chambers that were automatically assigned to the storage pool and reassign them to the entry/exit pool to provide additional chambers for importing and exporting magazines.
- This command does not enable encryption. If you want to use encryption with the partition, modify the partition after it is created to enable encryption. See the *Spectra Encryption User Guide* for detailed instructions.

Syntax

```
partition.xml?action=autoCreate&
  partition=[partition name]&
  saveLibraryConfiguration=[USB|[emailRecipient]]
  [&extraInformation=[string]]
```

where the value for:

This parameter...	Specifies...
partition	The name you want to use for the partition. Names can be any length and can include @ ?] _ /. and the space character. Partition names over 32 characters cause a scroll bar to display on some screens and are not recommended. **Note:** The partition name is case-sensitive.

This parameter...	Specifies...
saveLibrary Configuration (optional)	Where you want to save the configuration backup file that the library generates after the partition is created. Values: ▪ **USB** = Saves the file to a USB device that is connected to the LCM. ▪ ***emailRecipient*** = The email address of an already-configured mail recipient to whom the library will email the configuration backup file. **Notes:** ▪ If you want to save the configuration backup file that the library generates after the partition is created to a USB device, make sure that the USB device is connected to the LCM before running the command. ▪ Do not send the configuration backup file to *autosupport@spectralogic.com*. Spectra Logic does not save emailed configuration files unless they are specifically requested for troubleshooting. ▪ See "Configure Mail Users" in you library *User Guide* for information about configuring mail users.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<partition>
   <status>OK</status>
   <message>[message text]</message>
</library>
```

Progress Use `partition.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
partition.xml?action=autocreate&partition=Partition 1&
   saveLibraryConfiguration=JaneSuperuser@YourCompany.com
```

immediately returns the following XML-formatted data:

```
<partition>
   <status>OK</status>
   <message>
      Started automatic partition creation. Set progress in your
      query for status.
   </message>
</library>
```

and creates a single storage partition named **Partition 1** and emails the configuration backup file to the mail user **JaneSuperuser@YourCompany.com.** When the command completes successfully, the response to `partition.xml?progress` contains `<status>OK</status>`.

delete

Description Deletes the specified partition from the library. After you delete a partition, you can reassign the drives and chambers previously assigned to that partition to an existing partition, or use them in a new partition.

Preparation Before deleting an existing partition, make sure you address the following:

- Spectra Logic strongly recommends backing up the library configuration, either to a USB device or as an attachment to an email sent to a previously configured mail recipient, before you delete a partition.
- To ensure that you do not inadvertently mix cartridges from one storage partition with those from another, use your storage management software to eject all of the cartridges from the partition you want to delete. The library moves the media to the partition's entry/exit pool. Export the media from the library as described in mediaExchange.xml on page 101 or "Exporting or Exchanging Magazines and Cartridges" in your library *User Guide*.

Important After you delete the partition, you cannot access any magazines in the chambers that were assigned to the partition's storage and entry/exit pools until you assign the chambers to another partition.

- If the storage partition configuration includes encryption, make sure that you export the encryption key(s) for any cartridges that are in the partition as described in the *Spectra Encryption User Guide*. You will need the encryption key(s) in order to access the data on the cartridges if you import the cartridges into another partition.
- If you plan to delete a cleaning partition, use the BlueScale user interface to edit any storage partitions that use the cleaning partition to unassociate the cleaning partition from the storage partition.

Syntax

```
partition.xml?action=delete&partition=[partition name]&
    saveLibraryConfiguration=[USB|[emailRecipient]]
    [&extraInformation=[string]]
```

where the value for:

This parameter...	Specifies...
partition	The name of the partition you want to delete. **Notes:** ▪ The partition name is case-sensitive. ▪ The partition name is set when the partition is created. ▪ Use the **partitionList.xml** command to retrieve a list of all the partitions currently configured in the library (see partitionList.xml on page 171).

This parameter...	Specifies...
saveLibrary Configuration	Where you want to save the configuration backup file that the library generates after it deletes the partition. Values: ▪ **USB** = Saves the file to a USB device connected to the LCM. ▪ ***emailRecipient*** = The email address of an already-configured mail recipient to whom the library will email the configuration backup file. **Notes:** ▪ Do not send the configuration backup file to *autosupport@spectralogic.com*. Spectra Logic does not save emailed configuration files unless they are specifically requested for troubleshooting. ▪ If you want to save the configuration backup file that the library generates after the partition is deleted to a USB device, connect the USB device to the LCM before running the command. ▪ See "Configure Mail Users" in you library *User Guide* for information about configuring mail users.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<library>
   <status>OK</status>
   <message>[message text]</message>
</library>
```

Progress Use `partition.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
partition.xml?action=delete&partition=Partition 1&
   saveLibraryConfiguration=JaneSuperuser@YourCompany.com
```

immediately returns the following XML-formatted data:

```
<library>
   <status>OK</status>
   <message>Started delete partition action. Set progress in
      your query for status.</message>
</library>
```

deletes the partition named **Partition 1,** and emails the configuration backup file to the mail user **JaneSuperuser@YourCompany.com**. When the command completes successfully, response to `partition.xml?progress` contains `<status>OK</status>`.

list

Description Lists all existing partitions including details such as partition type, size, assigned drives, etc.

Note: This action was added with BlueScale12.5.0.

Syntax `partition.xml?action=list`

Command Response The command returns the following XML-formatted data:

```
<partition>
   <partitionData>
      <name>[partition name]</name>
      <type>
         LTO|LTO Cleaning|TS11x0|TS11x0 Cleaning|T10K
      </type>
      <emulation>[emulation type]</emulation>
      <includeDriveAndMediaGenerationInRES>
         yes|no
      </includeDriveAndMediaGenerationInRES>
      <softLoad>yes|no</softLoad>
      <mediaZoning>yes|no</mediaZoning>
      <barcodeLength>1-16</barcodeLength>
      <barcodeShortenedOnLeft>yes|no</barcodeShortenedOnLeft>
      <barcodeChecksum>yes|no</barcodeChecksum>
      <barcodeChecksumCalculated>
         yes|no
      </barcodeChecksumCalculated>
      <exporters>
         <exporter>
            <ID>[ID]</ID>
            <type>drive|QIP|unknown</type>
            <port>A|B|AB</port>
         </exporter>
         …
      </exporters>
      <slotsPerChamber>[value]</slotsPerChamber>
      <numStorageSlots>[value]</numStorageSlots>
      <numEESlots>[value]</numEESlots>
      <eeType>standard|queued|shared</eeType>
      <drives>
         <ID>[ID]</ID>
         …
      </drives>
      <globalSpares>
         <ID>[ID]</ID>
         …
      </globalSpares>
```

```
            <MLMMediaVerification>
               <preScan>yes|no</preScan>
               <postScan>
                  none|full|quickWithGlobalSpareDrives|
                  quickWithInlineDrives
               </postScan>
               <scanAfterDays>yes|no</scanAfterDays>
               <scanAfterWrite>yes|no</scanAfterWrite>
               <scanAfterRead>yes|no</scanAfterRead>
               <daysToScanAfter>[value]</daysToScanAfter>
            </MLMMediaVerification>
            <allowUsers>all|listOnly</allowUsers>
            <userList>
               <name>[user name]</name>
               …
            </userList>

            <affectedQIPs>
               <QIP>
                  <ID>[ID]</ID>
                  <exportedDrive>
                     <ID>[ID]</ID>
                     <port>A|B|AB</port>
                  </exportedDrive>
                  …
               </QIP>
               …
            </affectedQIPs>
            <cleaningPartition>[partition name]</cleaningPartition>
            <encryption>
               <type>
                  QIPBasedBlueScale|DriveBasedBlueScale|SpectraSKLM|KMIP
               </type>
               <encryptionKey>[moniker]</encryptionKey>
               <decryptionKeys>
                  <moniker>[moniker]</moniker>
                  …
               </decryptionKeys>
            </encryption>
            <moveOptionType>default|slotIQ</moveOptionType>
         </partitionData>
         …
</partition>
```

where the value for:

This parameter...	Specifies...
partitionData	The beginning of data for one partition.
name	The name of the partition. The partition name is case-sensitive.
type	The type of partition. Values: **LTO**, **LTO Cleaning**, **TS11x0**, **TS11x0 Cleaning**, **T10K** **Note:** The only parameters that will show for cleaning partitions are: **name, type, slotsPerChamber**, and **numStorageSlots**.
emulation	The emulation type, if any, that is applied to the partition. **Note:** Emulation will only appear if it is not the default "SPECTRA PYTHON"

This parameter...	Specifies...
includeDriveAndMediaGenerationInRES	Whether the response to the SCSI Read Element Status command will include drive and media generation data. Values: **yes, no**
softLoad	Whether the partition uses soft load. Values: **yes, no**
mediaZoning	That zone information is included in the SCSI Read Element Status command response. Values: **yes**, **no** **Note:** This parameter only applies to TFinity libraries.
barcodeLength	The number of barcode characters, including the checksum if applicable, the library reports. Values: 1-16. **Note:** This parameter only applies to TFinity libraries.
barcodeShortenedOnLeft	If the reported barcode is shortened, whether left side characters are removed. For example, if the barcode is 1234567L2, **barcodeShortenedOnLeft=no**, and **barcodeLength=5**, the library reports the barcode as 12345. If **barcodShortenedOnLeft=yes**, and **barcodeLength=5** the library reports the barcode as 567L2. Values: **yes**, **no** **Note:** This parameter only applies to TFinity libraries.
barcodeChecksum	Whether the barcode labels include a checksum character. Values: **yes**, **no** **Note:** This parameter only applies to TFinity libraries.
barcodeChecksumCalculated	Whether the barcode is verified against the checksum when it is read. Values: **yes**, **no** **Note:** This parameter only applies to TFinity libraries.
exporters	The name, type, and addressing for the controller(s) (RIM, F-QIP) or direct-attach drive(s) that provide the robotic control path for the partition. ▪ **ID** ▪ For a RIM or QIP, **ID** = The component identifier for the exporting RIM or F-QIP using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*].** See, QIPExporter on page 155 for a description of the parameters in the component identifier. ▪ For a direct-attached drive, **ID** = The component identifier for the exporting drive(s) using the form **DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*].** See driveExporter on page 157 for a description of the parameters in the component identifier. **Note:** See "Component Identifiers" in your library *User Guide* for more information, including ranges for each part of the identifier. ▪ **type** describes the type of device being used to provide the robotic control path. Values: **drive, QIP, unknown.** ▪ **port** is the port of the exporting controller that provides the robotic control path for the partition. Values: **A**, **B**, **AB.** **Note:** Port information will only appear if the exporter is an F-QIP or RIM.

This parameter...	Specifies...
slotsPerChamber	The number of slots per magazine. This number depends on the technology. Each LTO magazine has slots for 10 LTO cartridges; a TS11*xx* technology magazine has 9 slots; a T10K magazine has 9 slots. For a T120 library, slotsPerChamber is always 1.
numStorageSlots	The number of slots assigned for storing data cartridges. **Note:** For libraries that use TeraPack magazines, the number of slots assigned for storage is a multiple of the number of slots per magazine. For an LTO partition, the number of slots must be a multiple of 10; for a TS11*xx* technology partition, the number of slots must be a multiple of 9; for a T10K partition, the number of slots must be a multiple of 9.
numEESlots	The number of slots assigned for the partition's entry/exit pool. **Notes:** ▪ For libraries that use TeraPack magazines, the number of slots assigned to the entry/exit pool is a multiple of the number of slots per magazine. For an LTO partition, the number of slots must be a multiple of 10; for a TS11*xx* technology partition, the number of slots must be a multiple of 9; for a T10K partition, the number of slots must be a multiple of 9. ▪ This parameter is not applicable for T120 libraries.
eeType	The E/E (Entry/Exit) port operation mode. Values: **standard**, **queued**, **shared** **Note:** The **queued** and **shared** values are only supported by the T120 library.
drives	List of the component identifiers for the drives assigned to the partition. **ID** = The component identifier for the drive(s) using the form **DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*].** See driveExporter on page 157 for a description of the parameters in the component identifier. **Notes:** ▪ If a drive is an exporter, it is only listed in the exporter section, see exporters, above. ▪ All drives must use the same technology (LTO, TS11*xx* technology, or T10K).
globalSpares	The component identifiers for the drives used as Global Spare drives for the partition using the form **FR[*integer*]/DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*]**. See driveExporter on page 157 for a description of the parameters in the component identifier.

This parameter...	Specifies...
MLM Media Verification	The MLM media verification features that are enabled for the partition. See "Configuring and Using Media Lifecycle Management" in your *User Guide* for more information. ▪ **preScan** whether or not the PreScan feature is enabled in the partition. Values: **yes**, **no.** ▪ **postScan** the type of PostScan operation enabled in the partition. Values: **none, full, quickWithGlobalSpareDrives, quickWithInlineDrives.** ▪ **scanAfterDays** whether or not a cartridge is added to the PostScan queue based on the number of days since the last scan. Values: **yes, no.** ▪ **scanAfterRead** whether or not a cartridge is added to the PostScan queue each time data is read from it. Values: **yes, no.** ▪ **scanAfterWrite** whether or not a cartridge is added to the PostScan queue each time data is written to it. Values: **yes, no.** ▪ **daysToScanAfter** the number of days to use for determining when to add a cartridge to the PostScan queue if **scanAfterDays=yes**.
allowUsers	Whether all users, or a specific list of users, are allowed to access the partition. Values: **all, listOnly.**
userList	Names of users allowed to access the partition if **allowUsers=listOnly**.
affectedQIPs	The name and addressing for each F-QIP that provides Fibre Channel connectivity for SCSI tape drives installed in the library. ▪ **ID** is the component identifier for the QIP using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]**. See, QIPExporter on page 155 for a description of the parameters in the component identifier. ▪ **Port** is the port of the QIP used to provide Fibre Channel connectivity. Values: **A**, **B**, **AB.**
cleaningPartition	The name of the cleaning partition assigned to the storage partition. This parameter will not display if no cleaning partition is assigned. The cleaning partition name is case-sensitive.
encryption	**Note:** The type of encryption, if any, enabled in the partition. ▪ **type** is the type of encryption used, BlueScale encryption using a QIP, BlueScale encryption using an encryption-enabled drive, Spectra SKLM encryption key management, or KMIP encryption key management. Values: **QIPBasedBlueScale, DriveBasedBlueScale, SpectraSKLM, KMIP**. ▪ **encryptionKey** is the moniker of the encryption key assigned to a partition if **type=QIPBasedBlueScale** or **DriveBasedBlueScale.** ▪ **decryptionKey** is the list of monikers for the decryption keys assigned to a partition if **type=QIPBasedBlueScale** or **DriveBasedBlueScale**. **Note:** A maximum of eight decryption keys can be assigned to a single partition.
moveOptionType	Whether or not the partition is configured to use SlotIQ. Values: **default, slotIQ**. **Notes:** ▪ This parameter only applies to TFinity libraries. ▪ SlotIQ cannot be enabled if the advanced utility Tape Barcode Verification is enabled. See displayTapeBarcodeVerificationSetting on page 215.

Example Command and Response The following command:

```
partition.xml?action=list
```

returns the following data for a T120 library with a single LTO storage partition named Partition 1, 100 storage slots, 8 entry/exit slots, the robotic control path provided by the only drive in the library, no encryption enabled, QuickScan with drives in the partition enabled, no global spare, and all users allowed to access it.

```
<partition>
   <partitionData>
      <name>Partition 1</name>
      <type>LTO</type>
      <numStorageSlots>100</numStorageSlots>
      <numEESlots>8</numEESlots>
      <eeType>standard</eeType>
      <includeDriveAndMediaGenerationInRES>
         no
      </includeDriveAndMediaGenerationInRES>
      <exporters>
         <exporter>
            <type>drive</type>
            <ID>sLTO-DRV5A</ID>
         </exporter>
      </exporters>
      <drives> </drives>
      <globalSpares> </globalSpares>
      <MLMMediaVerification>
         <preScan>no</preScan>
         <postScan>quickWithInlineDrives</postScan>
         <scanAfterDays>no</scanAfterDays>
         <scanAfterWrite>no</scanAfterWrite>
         <scanAfterRead>no</scanAfterRead>
         <daysToScanAfter>0</daysToScanAfter>
      </MLMMediaVerification>
      <allowUsers>all</allowUsers>
   </partitionData>
</partition>
```

new

Description Creates a storage partition or cleaning partition that uses a specified number of slots, drives, and other configuration parameters specified by the command. The command configures the partition to allow access by all users. If you want to limit the users allowed to access the partition, modify the partition using the BlueScale interface. See your *User Guide* for details.

Requirements and Guidelines When preparing to use this command, keep the following requirements and guidelines in mind. Refer to the "Configuring and Managing Partitions" chapter in your library *User Guide* for detailed requirements and guidelines for configuring both data and cleaning partitions.

- Only one partition can be configured with each command.
- When configuring a storage partition, all drives and controllers (RIM or F-QIP) specified in the command must already be installed in the library. If you specify a device that is not present or that is impaired, the command fails.
- For libraries using a direct-attached drive to provide the robotic control path for a storage partition, the exporting drive counts as the one drive required to create the partition.
- A cleaning partition can be shared by multiple storage partitions as long as the cleaning cartridges are compatible with the drive types in the storage partitions.
- This command does not enable encryption. If you want to use encryption with the partition, modify the partition after it is created to enable encryption. See the *Spectra Encryption User Guide* for detailed instructions.
- **Restrictions on the Number of Partitions** — By default, the library lets you create a single storage partition. If you need additional storage partitions, you must purchase a Shared Library Services (SLS) activation key.
  - Libraries running BlueScale software versions earlier than BlueScale12.6.3, are limited to eight storage partitions.
  - Libraries running BlueScale12.6.3 or later can have a maximum of 16 storage partitions.

  Notes:
  - Cleaning partitions do not require an SLS activation key and do not count toward the partition maximum.
  - The more partitions in a library, the longer each move can take. If move requests can be sent to several partitions at once, you may need to increase the timeout setting in your storage management software.
- **Restrictions on Exporters** — Each partition is exported by a QIP/RIM, or a Fibre Channel or SAS drive. See your library *User Guide* for restrictions on the type and number of exporters.

Syntax

Notes:
- To simplify locating the corresponding information in your library *User Guide*, the parameters in the following syntax statement are listed in the same order as the corresponding settings presented by the BlueScale partition wizard. In some cases, the command parameters combine the settings from multiple BlueScale partition wizard screens.
- The parameters following the initial **action=new** parameter do not need to be entered in any specific order.

```
partition.xml?action=new&partition=[partition name]&
   type=[LTO|LTO Cleaning|TS11x0|TS11x0 Cleaning|T10K]&
   QIPExporter=[QIP ID][;robot visibility[:[addressing mode
      [:[hardAddress]]][;robot visibility[:[addressing mode
      [:[hardAddress]]]],...,[QIP ID][;robot visibility
      [:[addressing mode[:[hardAddress]]][;robot visibility
      [:[addressing mode[:[hardAddress]]]]&
   driveExporter=[Drive ID]:[address],...,[Drive ID]:[address]&
   globalSpares=[First Drive ID],...,[Last Drive ID]&
   numStorageSlots=[value]&numEESlots=[value]&
   eeType=[standard|queued|shared]&barcodeLength=<1-16>&
   barcodeShortenedSide=[left|right]&barcodeChecksum=[yes|no]&
   barcodeChecksumCalculated=[yes|no]&
   drives=[First Drive ID]:[portaddress],...,
      [Last Drive ID]:[portaddress]&
   cleaningPartition=[partition name]&enablePrescan&
   enableFullscan&
   enableQuickScan=[inlineDrives|globalSpareDrives]&
   scanAfter=[time:value|write|read],...,[time:value|write|read]&
   includeDriveAndMediaGenerationInRES&enableSoftLoad&slotIQ&
   enableMediaZoning&
   QIPList=[First QIP ID][;drive visibility[:[addressing mode
      [:[hardAddress]]][;drive visibility[:[addressing mode
      [:[hardAddress]]]],
      ...,
      [Last QIP ID][;drive visibility[:[addressing mode
      [:[hardAddress]]][;drive visibility[:[addressing mode
      [:[hardAddress]]]]&
   saveLibraryConfiguration=[USB|[emailRecipient]]
   [&extraInformation=[string]]
```

where the value for:

This parameter...	Specifies...
partition	The name you want to use for the partition. Names can be any length and can include @ ?] _ /. and the space character. Partition names over 32 characters cause a scroll bar to display on some screens and are not recommended. **Important:** The partition name is case-sensitive.

This parameter...	Specifies...
type	The type of partition to create. Values: **LTO**, **LTO Cleaning**, **TS11x0**, **TS11x0 Cleaning**, **T10K** **Important:** If you specify a drive type that is not present, the command fails. **Notes:** ▪ The **TS11x0** and **TS11x0 Cleaning** parameter values are only valid in a T380, T950, or TFinity library that has TS11*xx* technology drives installed. ▪ The **T10K** parameter value is only valid in TFinity libraries that have T10K drives installed.
QIPExporter	The name and addressing for the controller(s) (RIM or F-QIP) that provide the robotic control path for the partition. These controllers "export" the partition to the hosts using the library, receiving and processing the robotic motion commands sent from the host to the transporter. **Important:** If the partition will use a direct-attached drive to provide the robotic control path, do not use the **QIPExporter** parameter. The library does not support selecting both drives and RIMs, to export the same partition. If both the **QIPExporter** and the **driveExporter** parameters are used in the same command, the command fails. **Important:** TS11*xx* technology partitions must use a QIPExporter and not a driveExporter. **Important:** T10K partitions must use a QIPExporter and not a driveExporter. **Important:** The maximum number of exporting devices (drives and controllers) supported by the library is six, however, each controller can export more than one partition. **Important:** The specified RIM or F-QIP cannot be specified as a failover partner for the controller failover feature. See "Choose the Robotic Control Path" in you library *User Guide* for additional requirements when using the controller failover feature. **Important:** A RIM or an F-QIP can be the exporting controller for multiple partitions. If this is the case, the configuration settings included in the command will affect all partitions associated with the exporting controller. **Important:** You can select multiple controllers to export the same partition. ▪ For the TFinity library, the library will accept up to six moves at a time from the data storage software and use the library's MediaIQ software to analyze the move queue and process the moves in the most efficient way. ▪ For all other libraries, your storage management software must manage the multiple paths. Multiple paths cannot be used at the same time

This parameter...	Specifies...

QIPExporter *(continued)*	For multiple exporters, separate each set of ***QIP ID***, ***robot visibility***, and ***hardAddress*** with commas.
	▪ ***QIP ID*** = The component identifier for the exporting RIM or F-QIP using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]**, where:
	  ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the controller is in a library that supports multiple frames.
	  ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the controller. Not used in T120 library component identifiers.
	  ▪ **F-QIP[*integer*]** = The designator of the controller bay where the QIP is installed. For all libraries except the T120, the value of [***integer***] is always **1**. For the T120 library, the value of [***integer***] is either **1** or **2**.
	▪ ***robot visibility*** = The port of the exporting controller that provides the robotic control path for the partition. Values: **A**, **B**
	Important: At least one port must be specified. The first port is separated from the QIP ID by a semicolon (;). If both ports are used, the second port is separated from the first by a semicolon (;). For each port, optionally include the addressing mode and the hard address, if required.
	▪ ***addressing mode*** = The Fibre mode each port on the controller will use. Only one mode can be specified in the command. If not included, the controller retains previous configuration settings. If not included and the controller was not previously configured, soft addressing is used. Values: **loop**, **fabric**, **auto**, where:
	  ▪ **loop** = Specifies the arbitrated loop addressing mode. The loop ID is set by the value of the ***hardAddress*** parameter.
	  ▪ **fabric** = Specifies the fabric addressing mode.
	  ▪ **auto** = Specifies that the addressing mode is auto-negotiated by the controller.
	▪ ***hardAddress*** = The fixed address assigned to the port when the addressing mode is either **loop** or **auto**. Values: **0** through **125**
	Notes:
	▪ The **QIPExporter** parameter is omitted when configuring a cleaning partition.
	▪ If the ports were configured for another partition that uses the same controller, it is only necessary to indicate whether the partition will use Port A, Port B, or both (entered as **;A**, **;B**, or **;A;B**, respectively). Do not include the colon or the addressing parameters in the command.
	▪ See the "Architecture" and the "Configuring and Managing Partitions" chapters in your library *User Guide* for detailed information about controller component identifiers and port addressing. For libraries that support the controller failover feature, these chapters also include information about additional requirements when configuring this feature.
	EXAMPLE: Setting the **QIPExporter** parameter to **QIP1;A:loop:100** specifies that Port A of QIP1 will provide robot visibility for the partition. The specified port will use loop addressing and the hard address will be set to 100.

This parameter...	Specifies...

driveExporter	The name and addressing for the direct-attached drive or drives that provide the robotic control path for the partition using ADI. These drives "export" the partition to the hosts using the library, receiving and processing the robotic motion commands sent from the host to the transporter.
	Important: If the partition will use a RIM or an F-QIP to provide the robotic control path, do not use the **driveExporter** parameter. If both the **QIPExporter** and the **driveExporter** parameters are used, the command fails.
	Important: An ADI option activation key must be stored in the library. For T950 and TFinity libraries, using an LTO-5 or later generation tape drive for the robotic control path is supported by BlueScale 12.6.3 and later.
	Important: A TS11*xx* technology drive cannot be used as a **driveExporter**. TS11*xx* technology partitions must use a QIPExporter (RIM).
	Important: A T10K drive cannot be used as a **driveExporter**. T10K partitions must use a QIPExporter (RIM).
	Important: Do not use a drive that is listed in the **globalSpares** parameter.
	Important: You can select multiple LTO-5 and later generation drives as controllers, and export the same changer interface over the drives to provide redundancy, as long as your storage management software supports this. These multiple paths cannot be used at the same time.
	Important: Partitions with greater than 930 slots, or spanning more than one frame, should not use a **driveExporter** for the robotic control path. Using a **driveExporter** for larger partitions may cause an inventory operation to time out in your data storage software.
	For multiple drive exporters, separate each set of ***Drive ID*** and ***address*** setting with commas.
	▪ ***Drive ID*** = The component identifier for the exporting drive using the form **DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*]**, where:
	  ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the controller is in a library that supports multiple frames.
	  ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the controller. Not used in T120 library component identifiers.
	  ▪ **[*interface*]** = The interface used by the drive.
	  Values:
	  ▪ **f** = Fibre Channel
	  ▪ **s** = Serial Attached SCSI (SAS)
	  ▪ (blank) = SCSI
	  ▪ **[*technology*]** = The technology used by the drive.
	  Values: **LTO**
	  ▪ **DRV[*integer*]** = The designator of the drive bay in the DBA, as viewed from the back of the library. For all libraries except the T120, the value of ***x*** can be **1** through **4**. For the T120 library, the value of ***x*** can be **1** through **6** for full-height drives, and **1b** through **6a** for half-height drives.
	▪ ***address*** = The Fibre Channel address or SCSI ID for the drive.
	Values:
	  ▪ Fibre Channel drives: **0** through **125** for a fixed address or **None** to specify that the drive uses soft addressing.
	  ▪ Direct-attached SCSI drives: **0** through **15** for a wide SCSI bus.

This parameter...	Specifies...
driveExporter (*continued*)	**Notes:** ▪ The **ID** values returned by the **driveList.xml** command without any parameters are the component identifiers for the drives currently installed in the library (see driveList.xml on page 33). ▪ The **driveExporter** parameter is omitted when configuring a cleaning partition. ▪ If the drive has two Fibre Channel ports, only one port at a time can be used. The drive detects the first port to have an active connection and applies the address settings to that port. ▪ For direct-attached SCSI drives, the SCSI ID you specify must not be assigned to any other devices on the same SCSI bus. Assigning the same SCSI ID to multiple devices will cause communication problems on the bus. ▪ See the "Architecture" and the "Configuring and Managing Partitions" chapters in your library *User Guide* for detailed information about drive component identifiers and addressing.
globalSpares (optional)	The component identifiers for the drives that will be used as Global Spare drives for the partition using the form **DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*].** See driveExporter on page 157 for a description of the parameters in the component identifier. **Important:** Do not use the drive specified in the **driveExporter** parameter or any of the drives listed in the **drives** parameter or the command fails. **Notes:** ▪ The **globalSpares** parameter is omitted when configuring a cleaning partition. ▪ The **globalSpares** parameter is omitted if you do not want to configure one or more Global Spare drives for the partition. ▪ A Global Spare must be configured if you want to configure FullScan. ▪ The **globalSpares** parameter is not valid for SCSI drives. ▪ The command can contain a comma-separated list of Drive ID parameters to specify multiple Global Spare drives. ▪ The **ID** values returned by the **driveList.xml** command without any parameters are the component identifiers for the drives currently installed in the library (see driveList.xml on page 33). ▪ See the "Configuring and Managing Partitions" chapter in your library *User Guide* for detailed information about configuring Global Spare drives.

This parameter...	Specifies...

numStorageSlots	The number of slots to configure in the partition for storing the cartridges that are accessible to the host.
	Values:
	▪ If Thin Provisioning is not enabled in the library, select a number between **1** and the number of licensed slots in the library.
	▪ If Thin Provisioning is enabled, select a number between **1** and the maximum number of elements (storage slots + entry/exit slots + drives) allowed. The element limits are shown below:
	  ▪ T200, T380, T680 libraries: 1,000
	  ▪ T950 libraries: 20,000;
	  ▪ TFinity libraries: 60,000
	Important: If the library has Thin Provisioning enabled, a single F-QIP or RIM can export 60,000 elements. A RIM2 does not have this limit.
	Important: For libraries that use TeraPack magazines, the number of slots assigned for storage must be a multiple of the number of slots per magazine. For an LTO partition, the number of slots must be a multiple of 10; for a TS11*xx* technology partition, the number of slots must be a multiple of 9; for a T10K partition, the number of slots must be a multiple of 9.
	Important: If Thin Provisioning is enabled it is possible for the combined configured number of storage slots and entry/exit slots in all partitions to exceed the licensed or physically available number of slots.
	Important: If Thin Provisioning is not enabled:
	  ▪ If you plan to create multiple partitions, be sure to reserve enough chambers to configure the other partitions.
	  ▪ If you licensed all of the chambers in the library and want to use a cleaning partition, be sure to reserve enough chambers for the cleaning partition. If you did not license all of the chambers in the library, the unlicensed chambers are available for use in cleaning partitions.

This parameter...	Specifies...
numEESlots	The number of slots to use for the partition's entry/exit pool. Values: ▪ If Thin Provisioning is not enabled in the library, the number of E/E slots can be up to the number of slots licensed in the library minus the number of slots assigned to the storage pool. ▪ If Thin Provisioning is enabled, select up to the maximum number of elements (storage slots + entry/exit slots + drives) allowed. The element limits are shown below: ▪ T200, T380, T680 libraries: 1,000 ▪ T950 libraries: 20,000; ▪ TFinity libraries: 60,000 ▪ a single QIP or RIM: 60,000 ▪ a single RIM2: unlimited **Important:** For libraries that use TeraPack magazines, the number of slots assigned to the entry/exit pool must be a multiple of the number of slots per magazine. For an LTO partition, the number of slots must be a multiple of 10; for a TS11*xx* technology partition, the number of slots must be a multiple of 9; for a T10K partition, the number of slots must be a multiple of 9. **Important:** If you specify a number of slots that is greater than the licensed number of slots, the command fails. **Notes:** ▪ Although using an entry/exit pool for each storage partition is considered optional, an entry/exit pool with at least one chamber is required for some library operations and is strongly recommended. ▪ The **numEESlots** parameter is omitted when configuring a cleaning partition. ▪ The **numEESlots** parameter is not supported for the T120 library.
eeType	The E/E (Entry/Exit) port operation mode. Values: **standard**, **queued**, **shared** **Important:** If a T120 library has multiple partitions configured, the value for the **eeType** parameter must be either **queued** or **shared**. All partitions in a library must use the same value for the **eeType** parameter. **Notes:** ▪ The **eeType** parameter is omitted when configuring a cleaning partition. ▪ The **eeType** parameter is optional when configuring a storage partition. If it is not present, the default value of **standard** is used. ▪ The **queued** and **shared** parameter values are only supported by the T120 library. See "Allocate Slots and Tape Drives" in the *Spectra T120 Library User Guide* for a detailed explanation of how queued eject mode and shared mode operate.

This parameter...	Specifies...
barcodeLength (optional)	The number of barcode characters the library reports. If your labels include a checksum, the number you enter includes the checksum character. Values: 1-16. The default number of characters is 16. **Important:** Be careful when specifying the number of characters to report. You may end up with duplicate barcodes reported. For example, with **barcodeShortenedSide=right** and **barcodeLength=5**, barcodes 12345XXXL2 and 12345ABCL3 are both reported as 12345. **Note:** This parameter only applies to TFinity libraries.
barcodeShortened Side (optional)	Which side of the barcode is truncated if not all characters are reported. For example, if the barcode is 1234567L2 and you configure **barcodeShortenedSide=right**, and **barcodeLength=5**, the library reports the barcode as 12345. If **barcodeShortenedSide=left**, and **barcodeLength=5**, the library reports the barcode as 567L2. Values: **right** (default), **left** **Note:** This parameter only applies to TFinity libraries.
barcodeChecksum (optional)	Whether your barcode labels include a checksum character. Values: **yes** (default), **no** **Note:** This parameter only applies to TFinity libraries.
barcodeChecksum Calculated (optional)	Whether you want the barcode verified against the checksum when it is read. Values: **yes** (default), **no** **Note:** This parameter only applies to TFinity libraries.

This parameter...	Specifies...

drives	The name and addressing for the drives to be assigned to the partition. The command can contain a comma-separated list of parameters for multiple drives.
	Important: Do not use the drive specified in the **driveExporter** parameter or any of the drives listed in the **globalSpares** parameter or the command fails.
	Values:
	▪ ***Drive ID*** = The component identifier for the drive using the **DBA[*integer*]/[*interface*][*technology*]–DRV[*integer*].** See driveExporter on page 157 for a description of the parameters in the component identifier.
	▪ ***Port*** = The port on the F-QIP that provides Fibre Channel connectivity for SCSI drives. The drive can be visible to the host through Port A, Port B, or both on the F-QIP. Values: **A**, **B**, **AB**
	▪ ***Address*** = Fibre Channel address or SCSI ID for each direct-attached drive. Values:
	  ▪ Fibre Channel drives: **0** through **125** for a fixed address or **None** to specify that the drive uses soft addressing.
	  ▪ Direct-attached SCSI drives: **0** through **15** for a wide SCSI bus.
	Notes:
	▪ Omit the **drives** parameter when configuring a cleaning partition.
	▪ A drive assigned as the **driveExporter** is automatically included in the partition.
	▪ The **ID** values returned by the **driveList.xml** command are the component identifiers for the drives currently installed in the library (see driveList.xml on page 33).
	▪ The **port** parameter is only applicable to SCSI drives that use an F-QIP to provide Fibre Channel connectivity to the host. It is not supported for Fibre Channel, SAS, or direct-attached SCSI drives.
	▪ The **address** parameter is only applicable to Fibre Channel and direct-attached SCSI drives. It is not supported for SAS or F-QIP-attached SCSI drives.
	▪ If the drive has two Fibre Channel ports, only one port at a time can be used. The drive detects the first port to have an active connection and applies the address settings to that port.
	▪ For direct-attached SCSI drives, make sure that the SCSI ID you specify is not assigned to any other devices on the same SCSI bus. Assigning the same SCSI ID to multiple devices on the same SCSI bus will cause communication problems on the bus.
	▪ See the "Architecture" and the "Configuring and Managing Partitions" chapters in your library *User Guide* for detailed information about drive component identifiers and addressing.

cleaningPartition (optional)	The name of the cleaning partition that you want to associate with the storage partition in order to use the Auto Drive Clean feature. **Notes:** ▪ Omit the **cleaningPartition** parameter when configuring a cleaning partition. ▪ The partition name is case-sensitive. ▪ The specified cleaning partition must already be configured on the library. ▪ The specified cleaning partition must use the same media type as the storage partition.

This parameter...	Specifies...
enablePrescan (optional)	That the partition will use the PreScan feature to discover cartridges in a storage partition in place of the default Media Auto Discovery process. **Important:** This parameter is only valid when Media Lifecycle Management (MLM) is enabled for the library (see MLMSettings on page 115). **Notes:** ▪ Omit the **enablePrescan** parameter when configuring a cleaning partition. ▪ The **enablePrescan** parameter is optional for storage partitions. ▪ See "Using PreScan" in your library *User Guide* for information about this parameter.
enableFullscan (optional)	That the partition will use the FullScan feature to verify each data cartridge. FullScan uses a Global Spare drive assigned to the partition to verify all of the data on each cartridge. **Important:** This parameter is only valid when Media Lifecycle Management (MLM) is enabled for the library (see MLMSettings on page 115). **Important:** The command cannot contain both the **enableFullscan** and the **enableQuickscan** parameters or the command fails. **Notes:** ▪ Omit the **enableFullscan** parameter when configuring a cleaning partition. ▪ The **enableFullscan** parameter is optional for storage partitions. ▪ The **enableFullscan** parameter is only valid if the partition includes Global Spare drives configured using the **globalSpares** parameter. ▪ When you include the **enableFullscan** parameter in the command, you must specify one or more PostScan triggers using the **scanAfter** parameter. ▪ See "Using PostScan" in your library *User Guide* for information about this parameter.

This parameter...	Specifies...
enableQuickscan (optional)	That the partition will use the QuickScan feature to verify each data cartridge. QuickScan uses either one of the drives in the partition or a Global Spare drive assigned to the partition to verify the data on a single wrap, from the beginning of the tape (BOT) to the end of the wrap or the end of recorded data (EOD), whichever comes first. **Important:** This parameter is only valid when Media Lifecycle Management (MLM) is enabled for the library (see MLMSettings on page 115). **Important:** The command cannot contain both the **enableFullscan** and the **enableQuickscan** parameters or the command fails. Values: ▪ **inlineDrives** = Use one of the drives in the partition to verify the data on a single wrap of each cartridge. ▪ **globalSpareDrives** = Use a Global Spare drive assigned to the partition to verify the data on a single wrap of each cartridge. **Notes:** ▪ Omit **enableQuickscan** parameter when configuring a cleaning partition. ▪ The **enableQuickscan** parameter is optional for storage partitions. ▪ The **enableQuickscan** parameter is only valid if the partition uses LTO-5 or later generation drives or TS11*xx* technology drives. ▪ When you include the **enableQuickscan** parameter in the command, you must specify one or more PostScan triggers using the **scanAfter** parameter. ▪ The **globalSpareDrives** parameter value is only valid if the partition includes Global Spare drives configured using the **globalSpares** parameter. ▪ See "Using PostScan" in your library *User Guide* for information about this parameter.
scanAfter (optional)	The PostScan triggers used to start either a FullScan or QuickScan operation. **Important:** This parameter is only valid when Media Lifecycle Management (MLM) is enabled for the library (see MLMSettings on page 115). Values: ▪ **Time:*n*** = Where ***n*>0**, add the cartridges in the partition to the automatic PostScan queue when ***n*** days have passed since the last scan. ▪ **write** = Add a cartridge to the automatic PostScan queue each time data is written to it. ▪ **read** = Add a cartridge to the automatic PostScan queue each time data is read from it. **Notes:** ▪ Omit the **scanAfter** parameter when configuring a cleaning partition. ▪ Omit the **scanAfter** parameter if the command does not include either the **enableFullscan** or **enableQuickscan** parameter. ▪ The command can contain a comma-separated list of multiple triggers. ▪ See "Using PostScan" in your library *User Guide* for information about this parameter.
includeDriveAndMediaGenerationInRES (optional)	That Media Domain, Media Type, Drive Domain, and Drive Type are included in the SCSI Read Element Status command response.

This parameter...	Specifies...
enableSoftLoad (optional)	That the partition uses soft load. The soft load feature uses the drives soft load (or auto load) functionality to improve library performance. **Notes:** ▪ Soft Load should only be used when recommended by Spectra Logic Technical Support. ▪ Soft load requires that the library have high performance transporters and that the partition uses LTO-5 or later generation drives. ▪ Soft load is not compatible with Media Lifecycle Management or TS11*xx* technology drives. ▪ This parameter only applies to TFinity libraries.
slotIQ (optional)	That the partition uses SlotIQ, which optimizes performance by virtualizing the library inventory. **Notes:** ▪ SlotIQ is not compatible with Media Lifecycle Management, multiple exporters, or T10K drives. ▪ This parameter only applies to TFinity libraries.
enableMediaZoning (optional)	That zone information is included in the SCSI Read Element Status command response. **Notes:** ▪ Media Zoning should only be used when recommended by Spectra Logic Technical Support. ▪ Media Zoning is not compatible SlotIQ. ▪ Media Zoning is only compatible with RIM2 controllers. ▪ This parameter only applies to TFinity libraries.

This parameter...	Specifies...

QIPList	The name and addressing for each F-QIP that provides Fibre Channel connectivity for SCSI tape drives installed in the library. The command can contain a comma-separated list of parameters for multiple QIPs.
	Values:
	▪ ***QIP ID*** = The component identifier for the F-QIP using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]**. See, QIPExporter on page 155 for a description of the parameters in the component identifier.
	▪ ***drive visibility*** = The F-QIP port that provides the host connection to the drive. Values: **A**, **B**
	Important: At least one port must be specified for each QIP listed. The first port is separated from the QIP ID by a semicolon (;). If both ports of the QIP are used, the second port is separated from the first by a semicolon (;). For each port, optionally include the addressing mode and the hard address, if required. If more than one QIP is included, the QIPs are in a comma-separated list.
	▪ ***addressing mode*** = The Fibre mode each port on the controller will use. Only one mode can be specified in the command. If not included, the controller retains previous configuration settings. If not included and the controller was not previously configured, soft addressing is used. Values: **loop**, **fabric**, **auto**, where:
	  ▪ **loop** = Specifies the arbitrated loop addressing mode. The loop ID is set by the value of the ***hardAddress*** parameter.
	  ▪ **fabric** = Specifies the fabric addressing mode.
	  ▪ **auto** = Specifies that the addressing mode is auto-negotiated by the controller.
	▪ ***hardAddress*** = The fixed address assigned to the port when the addressing mode is either **loop** or **auto**. Values: **0** through **125**
	Notes:
	▪ Omit the **QIPList** parameter when configuring a cleaning partition.
	▪ Omit the **QIPList** parameter if there are no additional F-QIPs or if there are no SCSI drives in the library.
	▪ If the ports have previously been configured for another partition that uses the same controller, it is only necessary to indicate whether the partition will use Port A, Port B, or both (entered as **;A**, **;B**, or **;A;B**, respectively). You do not need to include the addressing parameters in the command unless you want to change the current settings.
	▪ Changing the configuration settings for an F-QIP port affects all partitions that use that F-QIP port.
	▪ See the "Architecture" and the "Configuring and Managing Partitions" chapters in your library *User Guide* for detailed information about component identifiers, port addressing, and drive visibility to the host.
	EXAMPLE: Setting the **QIPList** parameter to **QIP1;A;B,QIP2;A;B** specifies that both ports on two QIPs will be used to provide drive visibility. Because no configuration settings are included in the command, both QIPs will use previously configured settings for Port A and Port B.

This parameter...	Specifies...
saveLibrary Configuration (optional)	Where you want to save the configuration backup file that the library generates after creating the partition. Values: ▪ **USB** = Saves the file to a USB device connected to the LCM. ▪ ***emailRecipient*** = The email address of an already-configured mail recipient to whom the library will email the configuration backup file. **Notes:** ▪ See "Configure Mail Users" in you library *User Guide* for information about configuring mail users. ▪ Do not send the configuration backup file to *autosupport@spectralogic.com*. Spectra Logic does not save emailed configuration files unless they are specifically requested for troubleshooting. ▪ If you want to save the configuration backup file that the library generates after the partition is created to a USB device, make sure that the USB device is connected to the LCM before running the command.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<partition>
   <status>OK</status>
   <message>[message text]</message>
</library>
```

Progress Use `partition.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command creates a partition in a T950 library. The partition uses Fibre Channel LTO-5 drives and a RIM to provide the robotic control path for the partition.

```
partition.xml?action=new&partition=Partition 1&type=LTO&
   QIPExporter=FR1/DBA1/F-QIP1;A:fabric&
   globalSpares=FR1/DBA1/fLTO-DRV4&numStorageSlots=920&
   numEESlots=10&drives=FR1/DBA1/fLTO-DRV1:A,
      FR1/DBA1/fLTO-DRV2:A&cleaningPartition=LTO Clean&
   enablePrescan&enableQuickScan=globalSpareDrives&
   scanAfter=write,read&
   saveLibraryConfiguration=JaneSuperuser@YourCompany.com
```

The library immediately returns the following XML-formatted data:

```
<library>
   <status>OK</status>
   <message>Started partition creation. Set progress in your
      query for status.</message>
</library>
```

When the command completes successfully, the response to `partition.xml?progress` contains `<status>OK</status>`.

The partition has the following characteristics:

- The RIM uses the fabric addressing mode. The robotics are only visible to the host through Port A.
- Drive FR1/DBA1/fLTO-DRV4 is configured as a Global Spare drive.
- The partition has 920 slots assigned to the storage pool and ten slots assigned to the entry/exit pool.
- Drives FR1/DBA1/fLTO-DRV1 and FR1/DBA1/fLTO-DRV2 are assigned to the partition. The drives use soft addressing.
- A cleaning partition named LTO Clean is assigned to the storage partition, which enables the Auto Drive Clean feature.
- Both PreScan and QuickScan using Global Spares are enabled. The PostScan trigger is set to add each cartridge to the automatic PostScan queue after it is ejected from a drive following either a data write or data read operation.
- The configuration backup file generated after the partition is configured is sent to mail user JaneSuperuser@YourCompany.com.

resizeSlots

Description Decreases or increases the size of a partition's storage slots or EE slots.

Note: This action was added with BlueScale12.7.00.

Syntax `partition.xml?action=resizeSlots&partition=[partition name]&type=[EE|storage]&increase=[value]&decrease=[value]&saveLibraryConfiguration=[USB|[emailRecipient]][&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
partition	The name of the partition. **Important:** The partition name is case-sensitive.
type	The type of slots to resize. Values: **storage**, **EE**
increase	The number of slots by which to increase the slot count of the EE pool or storage pool. **Important:** For libraries that use TeraPack magazines, the number of slots must be a multiple of the number of slots per magazine. For an LTO partition, the number of slots must be a multiple of 10; for a TS11*xx* technology partition, the number of slots must be a multiple of 9; for a T10K partition, the number of slots must be a multiple of 9. **Important:** Partitions using a **driveExporter** for the robotic control path should not have the number of storage slots increased to be greater than 930 slots. Using a **driveExporter** for larger partitions may cause an inventory operation to time out in your data storage software. **Important:** If you specify a number of slots that makes the total number of slots assigned to partitions greater than the licensed number of slots, the command fails. **Important:** If you include both the increase and the decrease parameter, the command fails.
decrease	The number of slots by which to decrease the slot count of the EE pool or storage pool. **Important:** For libraries that use TeraPack magazines, the number of slots must be a multiple of the number of slots per magazine. For an LTO partition, the number of slots must be a multiple of 10; for a TS11*xx* technology partition, the number of slots must be a multiple of 9; for a T10K partition, the number of slots must be a multiple of 9. **Important:** If you include both the increase and the decrease parameter, the command fails.

This parameter...	Specifies...
saveLibrary Configuration (optional)	Where you want to save the configuration backup file that the library generates after creating the partition. Values: ▪ **USB** = Saves the file to a USB device connected to the LCM. ▪ ***emailRecipient*** = The email address of an already-configured mail recipient to whom the library will email the configuration backup file. **Notes:** ▪ See "Configure Mail Users" in you library *User Guide* for information about configuring mail users. ▪ Do not send the configuration backup file to *autosupport@spectralogic.com*. Spectra Logic does not save emailed configuration files unless they are specifically requested for troubleshooting. ▪ If you want to save the configuration backup file that the library generates after the partition is created to a USB device, make sure that the USB device is connected to the LCM before running the command.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<partition>
   <status>OK</status>
   <message>[message text]</message>
</library>
```

Progress Use `partition.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command decreases the size of partition Partition1 by 20 slots and saves the library configuration to a USB device.

```
partition.xml?action=resizeSlots&type=storage&partition=Partition1
&decrease=20&saveLibraryConfiguration=USB
```

The library immediately returns the following XML-formatted data:

```
<library>
   <status>OK</status>
   <message>Started partition resize. Set progress in your
      query for status.</message>
</library>
```

When the command completes successfully, the response to `partition.xml?progress` contains `<status>OK</status>`.

CHAPTER 19

partitionList

partitionList.xml

Use the **partitionList.xml** command to retrieve a list of all currently configured partitions.

[no parameters]

Description Returns a list of all the partitions configured in the library.

Syntax `partitionList.xml`

Command Response The command immediately returns the following XML-formatted data:

```
<library>
   <partitionName>[first partition name]</partitionName>
   ...
   <partitionName>[last partition name]</partitionName>
</library>
```

where the value for:

This parameter...	Specifies...
partitionName	The exact name of each partition configured in the library. **Note:** The partition name is set when the partition is created. See partition.xml on page 142 for information about using the XML command interface to configure partitions in the library.

Example Command and Response The following command:

```
partitionList.xml
```

immediately returns the following list of partitions:

```
<library>
   <partitionName>Partition 1</partitionName>
   <partitionName>Clean 1</partitionName>
</library>
```

CHAPTER 20

physInventory

physInventory.xml

Use the **physInventory.xml** command to retrieve a list of all occupied magazine and cartridge locations in the specified partition. The list includes the offset value for each occupied magazine and slot, as well as the barcodes of the magazines and cartridges, if available.

Note: Empty locations are not included in the list, but can be identified by the gaps in the offset values returned by the command.

partition

Syntax `physInventory.xml?partition=[partition name]`

where the value for:

This parameter...	Specifies...
partition	The exact name of the partition for which you want a physical inventory list. **Notes:** ▪ Use the **partitionList.xml** command to retrieve a list of all the partitions currently configured in the library (see partitionList.xml on page 171). Partition names are case sensitive. ▪ The partition name is set when the partition is created. See partition.xml on page 142 for information about using the XML command interface to configure partitions in the library.

Command Response The command immediately returns the following XML-formatted data:

```
<physInventory>
   <partition>
      <name>[paritition name]</name>
      <storage>
         <magazine>
            <offset>[value]</offset>
            <barcode>[value]</barcode>
            <frameNumber>[value]</frameNumber>
            <tapeBayNumber>[value]</tapeBayNumber>
            <drawerNumber>[value]</drawerNumber>
            <slot>
               <number>[value]</number>
               <barcode>[value]</barcode>
            </slot>
               ...
         </magazine>
         ...
      </storage>
      <entryExit>
         <magazine>
            <offset>[value]</offset>
            <barcode>[value]</barcode>
            <frameNumber>[value]</frameNumber>
            <tapeBayNumber>[value]</tapeBayNumber>
            <drawerNumber>[value]</drawerNumber>
            <slot>
               <number>[value]</number>
               <barcode>[value]</barcode>
               <barcodeValid>[Yes]</barcodeValid>
               </slot>
               ...
         </magazine>
         ...
      </entryExit>
   </partition>
</physInventory>
```

where the value for:

This parameter...	Indicates...
name	The name of the partition.
storage	The section of the XML data that contains all chambers in the storage pool of the partition.
entryExit	The section of the XML data that contains all chambers in the entry/exit pool of the partition.
magazine	The section of the XML data that contains a single magazine.
offset	The logical address of the magazine in the library. The logical addresses for magazines are sequential, beginning with **1**. **Important:** The **offset** values given by **physInventory.xml** are one-based. The **TeraPackOffsets** required by **mediaExchange.xml** are zero-based. You must subtract 1 from the **offset** values before supplying them as **TeraPackOffsets**. **Note:** Use the gaps in the values returned for the magazine offsets to identify the values that can be used for the **TeraPackOffset** parameter in the `mediaExchange.xml?action=importExport` command to import magazines (see importExport on page 106). **EXAMPLE:** The example command response on page 175 provides the following offset information: ▪ The offset for the first magazine listed is **1**. There are no slot parameters associated with this magazine, indicating that the magazine is empty. ▪ The offset for the second magazine listed is **6**. This magazine has two sets of slot data, indicating that there are cartridges in slots **1** and **3** of the magazine and that the remaining slots in the magazine are empty. ▪ Offset values **2**, **3**, **4**, and **5** are not included in the data, indicating the these locations are empty.
barcode	The barcode label information (magazine or cartridge). **Note:** If a storage slot does not contain a cartridge, or a chamber does not contain a magazine, the barcode information field is not returned.
frameNumber	The number of the frame where the magazine is located. **Note:** For libraries that do not support multiple frames, the value for the **frameNumber** parameter is always **1**.
tapeBayNumber	The number of the shelving bay within the specified frame where the magazine is located.
drawerNumber	The component identifier for the chamber (drawer) where the magazine is located. **Note:** This parameter is not supported by the T120 library.
slot	The section of the XML data that contains a single slot in a magazine.
number	The magazine slot number where the cartridge is located. **Notes:** ▪ If the slot does not contain a cartridge, the slot number field is not returned. ▪ This parameter is not returned for the T120 library.

Example Command and Response The following command:

```
physInventory.xml?partition=Partition 1
```

returns the following information about Partition 1:

```
<physInventory>
   <partition>
      <name>Partition 1</name>
      <storage>
         <magazine>
            <offset>1</offset>
            <barcode>LU83567</barcode>
            <frameNumber>1</frameNumber>
            <tapeBayNumber>1</tapeBayNumber>
            <drawerNumber>1</drawerNumber>
         </magazine>
         <magazine>
            <offset>6</offset>
            <barcode>LU57356</barcode>
            <frameNumber>1</frameNumber>
            <tapeBayNumber>1</tapeBayNumber>
            <drawerNumber>2</drawerNumber>
            <slot>
               <number>1</number>
               <barcode>000380L5</barcode>
            </slot>
            <slot>
               <number>3</number>
               <barcode>000862L4</barcode>
            </slot>
            ...
         </magazine>
         ...
         </magazine>
      </storage>
      <entryExit>
         <magazine>
            <offset>60</offset>
            <barcode>LU0591L4</barcode>
            <frameNumber>1</frameNumber>
            <tapeBayNumber>1</tapeBayNumber>
            <drawerNumber>3</drawerNumber>
         </magazine>
         <magazine>
            <offset>71</offset>
            <barcode>LU6847L4</barcode>
            <frameNumber>1</frameNumber>
            <tapeBayNumber>1</tapeBayNumber>
            <drawerNumber>4</drawerNumber>
         </magazine>
      </entryExit>
   </partition>
</physInventory>
```

USING THE PHYSINVENTORY.XML COMMAND

The following steps illustrate using the **physInventory.xml** command to retrieve the physical inventory of a partition named Partition 1. You can then use the data as parameter values for other XML commands (for example, in the **mediaExchange.xml** command described beginning on page 101).

1. Run the following command to retrieve a list of all occupied magazine and cartridge locations in the specified partition.

   ```
   physInventory.xml?partition=Partition 1
   ```

2. Depending on whether the magazine locations are in the partition's storage pool or entry/exit pool, locate the desired section of the physical inventory data.
   - The magazine locations in the storage pool are in the section of the data between **<storage>** and **</storage>**.
   - The magazine locations in the entry/exit pool are in the section of the data between <`entryExit`> and </`entryExit`>.
3. Identify the offset values for each magazine location you want to use.
   - If you plan to import magazines, identify gaps in the **offset** values in either the **storage** or **entryExit** sections of the physical inventory data. Each missing value corresponds to an empty location into which a magazine can be imported.
   - If you plan to export or exchange magazines, identify the magazines by examining the **barcode** data in the **magazine** and **slot** sections for each magazine in the desired section of the physical inventory data. For each magazine you want to export or exchange, determine its **offset** value.

Important The **offset** values given by **physInventory.xml** are one-based. The **TeraPackOffsets** required by **mediaExchange.xml** are zero-based. You must subtract 1 from the **offset** values before supplying them as **TeraPackOffsets**.

EXAMPLE The command response data for the **physInventory.xml** command shown on page 175, shows the first two magazine locations in the storage pool.

- The **storage** section of the physical inventory data does not include **offset** values **2**, **3**, **4**, or **5**. These missing offset values indicate these magazine locations are empty and can be used as destinations when importing magazines.
- The magazine with an **offset** value of **1** does not include a **slot** section, indicating that the magazine is empty.
- The magazine with an **offset** value of **6**, includes two **slot** sections, **number 1** and **number 3**, indicating that each of these slots contains a cartridge; there is not a **slot number 2**, indicating that this slot is empty.

CHAPTER 21

powerOff

powerOff.xml

Use the **powerOff.xml** command to power-cycles the library. See "Controlling the Library Power" in your library *User Guide* for detailed information about power-cycling the library.

[no parameters]

Description Powers the library off and then, if desired, powers it on again.

Preparation Before powering off the library, use the following steps to prepare for shut-down.

1. Use your storage management software to stop any file storage operations running to the library.
2. Pause PostScan if it is running (see "Pause the PostScan Process" in your library *User Guide*). Any tapes currently being scanned are returned to their storage locations.

Syntax `powerOff.xml?numSecondsToRemainOff=[seconds]`

where the value for:

This parameter...	Specifies...
numSecondsTo RemainOff	The number of seconds to wait after the **powerOff.xml** command completes before powering the library back on again. Values: ▪ **0** (zero) = The library remains off until powered on again from the operator panel. ▪ ***n*>0** = The library powers on again after the specified number of seconds. **Important:** There is no upper limit to the value for numSecondsToRemainOff. For example, if you enter 10000, the library remains off for approximately 2.8 hours. The only way to override the wait, once started, is to remove all power from the library (set all breakers to the off (down) position), wait 20 seconds, return power (set all breakers to the on (up) position), and press the front power button.

Command Response The command immediately returns the following XML-formatted data:

```
<powerOff>
   <status>OK</status>
</powerOff>
```

Example Command and Response The following command:

```
powerOff.xml?numSecondsToRemainOff=60
```

immediately returns the following XML-formatted data:

```
<powerOff>
   <status>OK</status>
</powerOff>
```

and powers off the library and then powers it back on again after 60 seconds.

CHAPTER 22

robotService

robotService.xml

Use the **robotService.xml** command to make a TFinity library send a robot to service or return a robot from service.

Notes:
- This command was added with BlueScale12.6.41.
- This command is only supported on TFinity libraries.

Action	
returnFromService	page 179
sendToService	page 180

returnFrom Service

Description Move a TFinity robot from the service bay to operation in the library. The service safety door must be open or the command fails.

Note: The command does not open the service safety door or instruct the operator to open the service safety door before the robot moves. If the door is closed the command fails.

Syntax

```
robotService.xml?action=returnFromService&robot=[1|2]
  [&extraInformation=[string]]
```

where the value for:

This parameter...	Indicates...
robot	The number of the robot in the library. Numbering is from left to right as viewed from the front of the library. Values: ▪ **1** = The left robot as viewed from the front of the library. ▪ **2** = The right robot as viewed from the front of the library.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command immediately returns the following XML-formatted data:

```
<robotService>
   <status>OK</status>
   <message>[message text]</message>
</robotService>
```

Progress Use `robotService.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
robotService.xml?action=returnFromService&robot=1
```

immediately returns the following XML-formatted data:

```
<robotService>
   <status>OK</status>
   <message>Started Robot Service Action. Set progress in your
      query for status.</message>
</robotService>
```

When the command completes successfully, the response to `robotService.xml?progress` contains `<status>OK</status>`.

sendTo Service

Description Move a TFinity robot to the service bay. The service safety door must be open or the command fails.

Notes:
- The command does not close the service safety door or instruct the operator to close the service safety door after the robot moves to service. Power to the robot remains on because the door is not closed.
- If at any time a motion restart occurs on the library, the robot reinitializes. If the robot was sent to the service bay, it will attempt to return to the library (return from service) on its own.

Syntax

```
robotService.xml?action=sendToService&robot=[1|2]
   [&extraInformation=[string]]
```

where the value for:

This parameter...	Indicates...
robot	The number of the robot in the library. Numbering is from left to right as viewed from the front of the library. Values: ▪ **1** = The left robot as viewed from the front of the library. ▪ **2** = The right robot as viewed from the front of the library.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command immediately returns the following XML-formatted data:

```
<robotService>
   <status>OK</status>
   <message>[message text]</message>
</robotService>
```

Progress Use `robotService.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
robotService.xml?action=sendToService&robot=1
```

immediately returns the following XML-formatted data:

```
<robotService>
   <status>OK</status>
   <message>Started Robot Service Action. Set progress in your
      query for status.</message>
</robotService>
```

When the command completes successfully, the response to `robotService.xml?progress` contains `<status>OK</status>`.

CHAPTER 23

robotUtilization

robotUtilization.xml

Use the **robotUtilization.xml** command to monitor the percentage of each hour that the library robotics are actively operating over the last 24 hours. The data is updated every hour while the library is powered on.

Notes:

- The command response can include up to 24 data sets, one for each hour that data was collected over a 24-hour period. The first time period begins one full hour after the library is powered on. Data collected during the first partial hour following power-on is discarded as invalid data.
- The data is stored in volatile memory and is not retained when the library is powered off.
- This command is not currently supported for the TFinity library.

[no parameters]

Description Returns the robotics usage data for the past 24 hour period, beginning with the most recent data.

Syntax `robotUtilization.xml`

Command Response The command immediately returns the following XML-formatted data:

```
<robotUtilization>
    <robotUtilizationDataPoint>
       <hourStartingAt>[most current hour]</hourStartingAt>
       <percentUtilization>[percent active]</percentUtilization>
    </robotUtilizationDataPoint>
...
    <robotUtilizationDataPoint>
       <hourStartingAt>[least current hour]</hourStartingAt>
       <percentUtilization>[percent active]</percentUtilization>
    </robotUtilizationDataPoint>
</robotUtilization>
```

where the value for:

This parameter...	Indicates...
robotUtilization DataPoint	The section of the XML data that contains a single usage data set.
hourStartingAt	The hour for which the usage data set was stored, using a 24-hour clock. Values: **1** through **24** **Note:** After 24 hours has elapsed, the data sets are deleted from the stored data on a First In, First Out (FIFO) basis.
percentUtilization	The integer value for the percentage of the hour during which the robot was active. Values: **0** through **100**, where **0** indicates that the robot was idle and **100** indicates that the robot was continuously performing moves during the hour.

Example Command and Response The following command:

```
robotUtilization.xml
```

returns the following XML-formatted data for the 24-hour period beginning at 4 PM (16:00 hours) and ending at 3 PM (15:00 hours) the following day:

```
<robotUtilization>
   <robotUtilizationDataPoint>
      <hourStartingAt>15</hourStartingAt>
      <percentUtilization>30</percentUtilization>
   </robotUtilizationDataPoint>
   ...
   <robotUtilizationDataPoint>
      <hourStartingAt>16</hourStartingAt>
      <percentUtilization>45</percentUtilization>
   </robotUtilizationDataPoint>
</robotUtilization>
```

CHAPTER 24

securityAudit

securityAudit.xml

Use the **securityAudit.xml** command to have a TFinity library check all TeraPack magazines to make sure that the expected tapes are in them.

Notes:
- This command is only supported on TFinity libraries.
- This command was added with BlueScale12.8.01.

abort

Description This command stops a currently in progress security audit.

Syntax `securityAudit.xml?action=abort`

Command Response The command immediately returns the following XML-formatted data:

```
<securityAudit>
   <status>OK</status>
   <message>[message text]</message>
</securityAudit>
```

Example Command and Response The following command:

```
securityAudit.xml?action=abort
```

immediately returns the following XML-formatted data:

```
<securityAudit>
   <status>OK</status>
   <message>Security audit aborted.</message>
</securityAudit>
```

start

Description Begin a security audit.

This command cannot run while another command that makes the robot(s) unavailable, such as verifyMagazineBarcodes on page 230, is running.

Syntax `securityAudit.xml?action=start`

Command Response The command immediately returns the following XML-formatted data:

```
<securityAudit>
   <status>OK</status>
   <message>[message text]</message>
</securityAudit>
```

Use `securityAudit.xml?status` to determine the progress of the operation. See status for details.

Example Command and Response The following command:

```
securityAudit.xml?action=start
```

immediately returns the following XML-formatted data:

```
<securityAudit>
   <status>OK</status>
   <message>Security audit started.</message>
</securityAudit>
```

Use `securityAudit.xml?action=status` to determine the progress of the operation. See status for details.

status

Description Determine the progress of a security audit.

Syntax `securityAudit.xml?action=status`

Command Response The command immediately returns the following XML-formatted data:

```
<securityAudit>
   <status>OK</status>
   <message>[message text]</message>
</securityAudit>
```

Example Command and Response The following command:

```
securityAudit.xml?action=status
```

immediately returns the following XML-formatted data:

```
<securityAudit>
   <status>OK</status>
   <message>Security audit is 52 percent complete.</message>
</securityAudit>
```

When the message is `"Security audit is not running."`, the previous audit is complete.

See getSecurityAuditLogNames on page 201, gatherSecurityAuditLog on page 203, and getSecurityAuditLog on page 204 for instructions for gathering the security audit log.

CHAPTER 25

systemMessages

systemMessages.xml

Use the **systemMessages.xml** command to retrieve all system messages currently stored on the library. See "Check and Respond to Messages" in your library *User Guide* for a detailed description of the library's messaging system.

[no parameters]

Description Returns the list of system messages that are currently stored on the library. The messages are listed in the order they were posted, beginning with the most recent.

Syntax `systemMessages.xml`

Command Response The command immediately returns the following XML-formatted data:

```
<systemMessages>
   <message>
      <number>[message number]</number>
      <severity>[Info|Warning|Error|Fatal Error]</severity>
         <date>
            <month>[mm]</month>
            <day>[dd]</day>
            <year>[yyyy]</year>
         </date>
         <time>
            <hour>[hh]</hour>
            <minute>[mm]</minute>
            <second>[ss]</second>
         </time>
         <notification>[message text]</notification>
         <remedy>[suggested remedy text]</remedy>
   </message>
</systemMessages>
```

where the value for:

This parameter...	Indicates...
number	The message number assigned by the library.
severity	The severity classification for the message. Values: ▪ **Info** = The library is working as intended. An event occurred that generated information about an operation or a system component. No action is required. ▪ **Warning** = The library is working as intended. An event occurred that may require attention to keep the library running at 100%. If the event was unexpected, determine the cause of the event and take remedial steps. ▪ **Error** = The library operation is impaired and requires user intervention. Examine any additional information in the message and take any required remedial steps. ▪ **Fatal Error** = The library has experienced an event that prevents it from continuing operations. Examine any additional information in the message and take any required remedial steps.
date	The month, day, and year (mm, dd, yyyy) that the system message was posted.
time	The hour, minute, and second (hh mm ss), using a 24-hour clock, that the system message was posted.
notification	The text displayed in the system message.
remedy	Any suggested remedy if the message indicates an error.

Example Command and Response The following command:

```
systemMessages.xml
```

retrieves the current system messages for the library. For example,

```
<systemMessages>
   <message>
      <number>32</number>
      <severity>Info</severity>
      <date>
         <month>6</month>
         <day>24</day>
         <year>2015</year>
      </date>
      <time>
         <hour>7</hour>
         <minute>36</minute>
         <second>40</second>
      </time>
      <notification>
         MLM Media discovery completed in partition Partition 1.
      </notification>
      <remedy>None.</remedy>
   </message>
   ...
   <message>
      <number>31</number>
      <severity>Warning</severity>
      <date>
         <month>6</month>
         <day>23</day>
         <year>2015</year>
      </date>
      <time>
         <hour>16</hour>
         <minute>19</minute>
         <second>08</second>
      </time>
      <notification>The tape with barcode 208139L4 has a tape
         health score of 49, and has reported a RED tape health the
         last 5 times it was loaded.</notification>
      <remedy>This could be a problem with the tape, or it could be
         a drive that is causing read or write errors. Investigate
         the media history of the tape with barcode 208139L4 in
         your MLM reports, and the health of the drive(s) that this
         tape has recently been loaded into. If the media itself is
         at fault, valid data on the tape should be migrated off of
         that media and the tape discarded to protect your data.
      </remedy>
   </message>
   ...
</systemMessages>
```

CHAPTER 26

taskList

taskList.xml

Use the **taskList.XML** command to retrieve a list of the operations currently in process on the library.

[no parameters]

Description Returns a list of the extended actions and background operations currently in process on the library. See Progress for Extended Action Commands on page 19 for additional information about how the library handles extended actions.

Syntax `taskList.xml`

Command Response

- If the library is not currently performing any extended or background operations, the command returns a page with empty parameter tags.
- If the library is performing any extended or background operations the command returns the following XML-formatted data:

```
<taskList>
   <currentAsynchronousAction>
      <name>[action name]</name>
      <status>
         QUEUE|SUBMITTED|ACTIVE|WAITINGFORUSER|DETACHED
      </status>
      <feedbackString>[feedback message]</feedbackString>
   </currentAsyncronousAction>
   <currentBackgroundTasks>
      <task>
         <name>[background task]</name>
         <thread>
            <description>[thread description]</description>
            <extraInformation>[string]</extraInformation>
         </thread>
      </task>
      ...
   </currentBackgroundTasks>
   <pageNeedingProgressRequest>
      [action name]
   </pageNeedingProgressRequest>
</taskList>
```

where the value for:

This parameter...	Indicates...
current Asynchronous Action	The section of the command response that contains information about any extended commands the library is processing. **Notes:** ▪ The library only processes one extended command at a time. ▪ If the library is not currently processing any extended commands, this section appears with empty parameter tags.
current BackgroundTask	The section of the command response that contains information about the background tasks the library is currently processing. **Notes:** ▪ The library can process multiple background tasks, most of them in parallel. ▪ If the library is not currently processing any background tasks, this section appears with empty parameter tags.
task	The section of the **currentBackgroundTask** section of the command response that contains information about each background task that is currently in process.
name	The name of the extended action or background task currently in process.
status	The current status of the extended action. Values: ▪ **QUEUE** = The action is waiting to run. ▪ **SUBMITTED** = The command was received. The command is in this state for a very short time. ▪ **ACTIVE** = The extended action is currently running. ▪ **WAITINGFORUSER** = The extended action is waiting for the operator to respond to the feedback string. ▪ **DETACHED** = The extended action was started by a user other than the one who issued the current **taskList.xml** command. The library is waiting for that action to complete before beginning the action indicated by the name parameter.
feedbackString	The text string that displays indicating that feedback is required from the operator before processing of the extended task can continue. **Note:** The **feedbackString** parameter is only included if the status is **WAITINGFORUSER**.
thread	Container for information about each background client involved in processing the background operation indicated by the **name** parameter in the task section of the response data.
description	A description of the background task.
extraInformation	An optional string added to the background command to help identify background tasks.
pageNeeding ProgressRequest	If applicable, the name of the extended action that needs a progress request issued in order to complete (see Progress for Extended Action Commands on page 19).

Example Command and Response The following command:

```
taskList.xml
```

provides the following response when a cartridge move is in process:

```
<taskList>
   <currentAsynchronousAction>
      <name>MOVE_ELEMENTS</name>
      <status>ACTIVE</status>
   </currentAsynchronousAction>
</taskList>
```

CHAPTER 27

traces

traces.xml

Use the **traces.xml** command to retrieve the CAN logs, QIP logs, and other traces stored on the library's LCM. In most cases, you only need to capture traces when instructed to do so by Spectra Logic Technical Support to assist in diagnosing problems with the library. See "Capturing Traces" in your library *User Guide* for additional information about the traces and logs.

getCanLogNames

Description Returns a list of the zip files containing the CAN logs that are currently stored in the LCM. The CAN logs collected for each day are zipped and stored on the hard drive in the LCM. Each zip filename includes the date it was created.

Important The **getCanLogNames** action is only supported for libraries that are using the Spectra LS module as the LCM. Issuing this command to a library that uses a Spectra PC as the LCM returns an empty list.

Syntax `traces.xml?action=getCanLogNames`

Command Response The command immediately returns the following XML-formatted data:

```
<traces>
   <canLogNames>
      <logName>latest.zip</logName>
      <logName>yy_mm_dd.zip</logName>
      ...
      <logName>yy_mm_dd.zip</logName>
   </canLogNames>
</traces>
```

where the value for:

This parameter...	Indicates...
logName	The name of the file containing the zipped set of CAN log files. Values: ▪ **latest.zip** = Contains the CAN log files for the current date. ▪ **yy_mm_dd.zip** = Contains the CAN log files that were generated on the date indicated in the filename, where **yy** is the last two digits of the year, **mm** is the two-digit month, and **dd** is the two-digit day.

Example Command and Response The following command:

```
traces.xml?action=getCanLogNames
```

returns the following XML-formatted data containing the list of CAN logs currently available.

```
<traces>
   <canLogNames>
      <logName>latest.zip</logName>
      <logName>12_03_25.zip</logName>
      ...
      <logName>11_11_20.zip</logName>
   </canLogNames>
</traces>
```

getCanLog

Description Retrieves the specified zip file containing CAN logs from the LCM. Use the `traces.xml?action=getCanLogNames` command (see getCanLogNames on page 193) to determine names of the zip files containing CAN logs that are currently stored on the hard drive in the LCM.

Important

The **getCanLog** action parameter is only supported for libraries that are using the Spectra LS module as the LCM. Issuing this command to a library that uses a Spectra PC as the LCM returns an empty list.

Syntax `traces.xml?action=getCanLog&name=[value]`

where the value for:

This parameter...	Indicates...
name	The name of the zipped set of log files being requested. Values: ▪ **latest.zip** = The CAN log files for the current date. The zip file is created at the time it is requested. ▪ **yy_mm_dd.zip** = The name of the zip file containing the CAN logs for a specific date, where **yy** is the last two digits of the year, **mm** is the two-digit month, and **dd** is the two-digit day.

Command Response The command immediately returns the requested CAN log files in a binary zip file.

Example Command The following command:

```
traces.xml?action=getCanLog&name=12_03_20.zip
```

returns a zip file containing the CAN logs that were collected on March 20, 2012.

getFullMotionLogNames

Description Returns a list of the motion logs on the RCM with an attribute to indicate which ones are also gathered and present on the LCM. The LCM keeps no more than five motion logs.

Notes:
- This action was added with BlueScale12.7.04.
- This action is only supported on TFinity libraries.

Syntax `traces.xml?action=getFullMotionLogNames`

Command Response The command immediately returns the following XML-formatted data:

```
<traces>
   <motionLogNames>
      <File>
         <logName>latest.tar</logName>
         <gathered>yes|no</gathered>
      </File>
      <File>
         <logName>
            motionSupport.serial_number.YYYY-MM-DDThhmm.tar
         </logName>
         <gathered>yes|no</gathered>
      </File>
      ...
   </motionLogNames>
</traces>
```

where the value for:

This parameter...	Indicates...
logName	The name of the tar file containing the motion log. Values: ▪ **latest.tar** = Contains the latest motion log. ▪ **motionSupport.serial_number.YYYY-MM-DDThhmm.tar** = Contains the motion log gathered from the library with the **serial number** shown at the time indicated in the filename, where **YYYY** is the year, **MM** is the two-digit month, **DD** is the two-digit day, **hh** is the 24-hour hour, and **mm** is the minute.
gathered	Whether the motion log is gathered on the LCM. Values: **yes**, **no**

Example Command and Response The following command:

```
traces.xml?action=getFullMotionLogNames
```

returns the following XML-formatted data containing the list of motion logs currently available.

```
<traces>
   <motionLogNames>
      <File>
         <logName>latest.tar</logName>
         <gathered>yes</gathered>
      </File>
      <File>
         <logName>
            motionSupport.0915A05.2017-07-30T2359.tar
         </logName>
         <gathered>no</gathered>
      </File>
      ...
   </motionLogNames>
</traces>
```

gatherFull MotionLog

Description Gathers the specified motion log to the LCM. See getFullMotionLogNames on page 194 to determine the names of the tar files currently stored on the RCM.

Notes:
- This action was added with BlueScale12.7.04.
- This action is only supported on TFinity libraries.

Syntax `traces.xml?action=gatherFullMotionLog&name=[value] [&extraInformation=[string]]`

where the value for:

This parameter...	Indicates...
name	The name of the tar file containing the motion log. Values: ▪ **latest.tar** = Contains the latest motion log. ▪ **motionSupport.serial_number.YYYY-MM-DDThhmm.tar** = Contains the motion log gathered from the library with the **serial number** shown at the time indicated in the filename, where **YYYY** is the year, **MM** is the two-digit month, **DD** is the two-digit day, **hh** is the 24-hour hour, and **mm** is the minute.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<traces>
   <status>OK</status>
   <message>[message text]</message>
</traces>
```

Example Command The following command:

```
traces.xml?action=gatherFullMotion&name=latest.tar
```

immediately returns the following XML-formatted data:

```
<traces>
   <status>OK</status>
   <message>Started Full Motion Log Gather. Set progress in your
   query for status.</message>
</traces>
```

When the motion log gather completes, the response to `traces.xml?progress` contains `<status>OK</status>`. Then use `traces.xml?action=getFullMotionLog&name=latest.tar` to retrieve the file.

getFullMotionLog

Description Retrieves the specified motion log from the LCM. See getFullMotionLogNames on page 194 to determine the names of the available tar files. If necessary, see gatherFullMotionLog on page 196 to move the file to the LCM.

Notes:
- This action was added with BlueScale12.7.04.
- This action is only supported on TFinity libraries.

Syntax `traces.xml?action=getFullMotionLog&name=[value]`

where the value for:

This parameter...	Indicates...
name	The name of the tar file containing the motion log. Values: ▪ **latest.tar** = Contains the latest motion log. **Note:** To make sure that you get a motion log gathered recently, the **gathered** flag resets to "no" when you get latest.tar. ▪ **motionSupport.serial_number.YYYY-MM-DDThhmm.tar** = Contains the motion log gathered from the library with the **serial number** shown at the time indicated in the filename, where **YYYY** is the year, **MM** is the two-digit month, **DD** is the two-digit day, **hh** is the 24-hour hour, and **mm** is the minute.

Command Response The command immediately returns the requested motion log.

Example Command The following command:

```
traces.xml?action=getFullMotionLog&name=latest.tar
```

returns the contents of the latest motion log and sets the gathered flag for latest.tar to "no".

getKernelLogNames

Description Returns a list of the zip files containing kernel logs that are currently stored in the LCM. The kernel logs generated each day are zipped and stored on the hard drive in the LCM. Each zip filename includes the date it was created.

Important

The **getKernelLogNames** action is only supported for libraries that are using the Spectra LS module as the LCM. Issuing this command to a library that uses a Spectra PC as the LCM returns an empty list.

Note: This action was added with BlueScale12.7.00.

Syntax `traces.xml?action=getKernelLogNames`

Command Response The command returns the following XML-formatted data:

```
<traces>
   <kernelLogNames>
      <logName>latest.zip</logName>
      <logName>yy_mm_dd.zip</logName>
      ...
      <logName>yy_mm_dd.zip</logName>
   </kernelLogNames>
</traces>
```

where the value for:

This parameter...	Indicates...
logName	The name of the file containing the zipped set of kernel log files. Values: ▪ **latest.zip** = Contains the kernel log files for the current date. ▪ **yy_mm_dd.zip** = Contains the kernel log files that were generated on the date indicated in the filename, where **yy** is the last two digits of the year, **mm** is the two-digit month, and **dd** is the two-digit day.

Example Command and Response The following command:

```
traces.xml?action=getKernelLogNames
```

retrieves a list of zip files containing kernel logs:

```
<traces>
   <kernelLogNames>

      <logName>latest.zip</logName>
      <logName>16_04_25.zip</logName>
      ...
      <logName>16_04_21.zip</logName>
   </kernelLogNames>
</traces>
```

getKernelLog

Description Retrieves the specified zip file containing kernel logs from the LCM. Use the `traces.xml?action=getKernelLogNames` command (see getKernelLogNames on page 197) to determine the names of the kernel logs currently stored on the hard drive in the LCM.

Important

The **getKernelLog** action is only supported for libraries that are using the Spectra LS module as the LCM. Issuing this command to a library that uses a Spectra PC as the LCM returns an empty list.

Note: This action was added with BlueScale12.7.00.

Syntax `traces.xml?action=getKernelLog&name=[value]`

where the value for:

This parameter...	Indicates...
name	The name of the zipped set of kernel log files being requested. Values: ▪ **latest.zip** = Contains the kernel log files for the current date. ▪ **yy_mm_dd.zip** = Contains the kernel log files that were generated on the date indicated in the filename, where **yy** is the last two digits of the year, **mm** is the two-digit month, and **dd** is the two-digit day.

Command Response The command immediately returns the requested kernel log files in a binary zip file.

Example Command The following command:

```
traces.xml?action=getKernelLog&name=16_04_25.zip
```

returns a zip file containing the kernel logs that were generated on April 25, 2016.

getQIPLogNames

Description Returns a list of the zip files containing QIP logs that are currently stored in the LCM. The QIP logs generated each day are zipped and stored on the hard drive in the LCM. Each zip filename includes the date it was created.

Important

The **getQIPLogNames** action is only supported for libraries that are using the Spectra LS module as the LCM. Issuing this command to a library that uses a Spectra PC as the LCM returns an empty list.

Syntax `traces.xml?action=getQIPLogNames`

Command Response The command returns the following XML-formatted data:

```
<traces>
   <qipLogNames>
      <logName>QIP_yy_mm_dd.zip</logName>
      ...
      <logName>QIP_yy_mm_dd.zip</logName>
   </qipLogNames>
</traces>
```

where the value for:

This parameter...	Indicates...
logName	The name of the file containing the zipped set of QIP log files. Values: **QIP_yy_mm_dd.zip**, where **QIP** identifies the file as a QIP log file, ***yy*** is the last two digits of the year, ***mm*** is the two-digit month, and ***dd*** is the two-digit day.

Example Command and Response The following command:

```
traces.xml?action=getQIPLogNames
```

retrieves a list of zip files containing QIP logs:

```
<traces>
   <qipLogNames>
      <logName>QIP_11_09_25.zip</logName>
      ...
      <logName>QIP_10_11_20.zip</logName>
   </qipLogNames>
</traces>
```

getQIPLog

Description Retrieves the specified zip file containing QIP logs from the LCM. Use the `traces.xml?action=getQIPLogNames` command (see getQIPLogNames on page 199) to determine the names of the QIP logs currently stored on the hard drive in the LCM.

Important The **getQIPLog** action is only supported for libraries that are using the Spectra LS module as the LCM. Issuing this command to a library that uses a Spectra PC as the LCM returns an empty list.

Syntax `traces.xml?action=getQIPLog&name=[value]`

where the value for:

This parameter...	Indicates...
name	The name of the zipped set of QIP log files being requested, using the form: **QIP**_*yy*_*mm*_*dd*.**zip**, where **QIP** identifies the file as a QIP log file, ***yy*** is the last two digits of the year, ***mm*** is the two-digit month, and ***dd*** is the two-digit day.

Command Response The command immediately returns the requested QIP log files in a binary zip file.

Example Command The following command:

```
traces.xml?action=getQIPLog&name=QIP_12_03_20.zip
```

returns a zip file containing the QIP logs that were generated on March 20, 2012.

getSecurityAuditLogNames

Description Returns a list of the security audit logs on the RCM with an attribute to indicate which ones are also gathered and present on the LCM. The library keeps at least the last five security audit logs on the RCM.

Notes:
- This action was added with BlueScale12.8.01.
- This action is only supported on TFinity libraries.

Syntax `traces.xml?action=getSecurityAuditLogNames`

Command Response The command immediately returns the following XML-formatted data:

```
<traces>
   <securityAuditLogNames>
      <File>
         <logName>
            securityAuditInterim.serial_number.YYYY-MM-
               DDThhmmss.sss.bz2
         </logName>
         <gathered>no</gathered>
      </File>
      <File>
         <logName>
            securityAudit.serial_number.YYYY-MM-DDThhmmss.sss.bz2
         </logName>
         <gathered>no</gathered>
      </File>
   </securityAuditLogNames>
</traces>
```

where the value for:

This parameter...	Indicates...
logName	The name of the file containing the security audit log. Values: ▪ **securityAuditInterim.serial_number.YYYY-MM-DDThhmmss.sss.bz2** contains the logs collected between security audits. This includes information such as doors being opened. ▪ **securityAudit.serial_number.YYYY-MM-DDThhmmss.sss.bz2** contains security audit logs. This includes information such as missing TeraPack magazines or tapes in unexpected slots. The names for both interim security audits and security audits contain the following information: ▪ **serial-number** shows the serial number of the library on which the security audit was run. ▪ **YYYY-MM-DDThhmmss.sss** for an interim security audit indicates the time and date the last audit ended and for a security audit indicates the time and date that the audit began. **YYYY** is the year, **MM** is the two-digit month, **DD** is the two-digit day, **hh** is the 24-hour hour, **mm** is the minute, and **ss.sss** is the seconds and milliseconds.
gathered	Whether the log is gathered on the LCM. Values: **yes**, **no**

Example Command and Response The following command:

```
traces.xml?action=securityAuditLogNames
```

returns the following XML-formatted data containing the list of security audit logs currently available.

```
<traces>
   <securityAuditLogNames>
      <File>
         <logName>
            securityAuditInterim.ZEPPAO1234.2019-03-
            05T215118.383.bz2
         </logName>
         <gathered>no</gathered>
      </File>
      <File>
         <logName>
            securityAudit.ZEPPAO1234.2019-03-05T152145.314.bz2
         </logName>
         <gathered>no</gathered>
      </File>
...
   </securityAuditLogNames>
</traces>
```

gather SecurityAudit Log

Description Gathers the specified security audit log to the LCM. See getSecurityAuditLogNames on page 201 to determine the names of the files currently stored on the RCM.

Notes:
- This action was added with BlueScale12.8.01.
- This action is only supported on TFinity libraries.

Syntax `traces.xml?action=gatherSecurityAuditLog&name=[value][&extraInformation=[string]]`

where the value for:

This parameter...	Indicates...
name	The name of the file containing the motion log. Values: ▪ **securityAuditInterim.serial_number.YYYY-MM-DDThhmmss.sss.bz2** contains the logs collected between security audits. This includes information such as doors being opened. ▪ **securityAudit.serial_number.YYYY-MM-DDThhmmss.sss.bz2** contains security audit logs. This includes information such as missing TeraPack magazines or tapes in unexpected slots. The names for both interim security audits and security audits contain the following information: ▪ **serial-number** shows the serial number of the library on which the security audit was run. ▪ **YYYY-MM-DDThhmmss.sss** for an interim security audit indicates the time and date the last audit ended and for a security audit indicates the time and date that the audit began. **YYYY** is the year, **MM** is the two-digit month, **DD** is the two-digit day, **hh** is the 24-hour hour, **mm** is the minute, and **ss.sss** is the seconds and milliseconds.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<traces>
   <status>OK</status>
   <message>[message text]</message>
</traces>
```

Example Command The following command:

```
traces.xml?action=gatherSecurityAuditLog&name=securityAudit.ZEPPAO
1234.2019-03-05T152145.314.bz2
```

immediately returns the following XML-formatted data:

```
<traces>
   <status>OK</status>
   <message>Started Security Audit Log Gather. Set progress in
      your query for status.</message>
</traces>
```

When the log gather completes, the response to `traces.xml?progress` contains `<status>OK</status>`. Then use `traces.xml?action=getSecurityAuditLog&name=[value]` to retrieve the file.

getSecurityAuditLog

Description Retrieves the specified security audit log from the LCM. See getSecurityAuditLogNames on page 201 to determine the names of the available files. If necessary, see gatherSecurityAuditLog on page 203 to move the file to the LCM.

Notes:
- This action was added with BlueScale12.8.01.
- This action is only supported on TFinity libraries.

Syntax `traces.xml?action=getSecurityAuditLog&name=[value]`

where the value for:

This parameter...	Indicates...
name	The name of the file containing the motion log. Values: ▪ **securityAuditInterim.serial_number.YYYY-MM-DDThhmmss.sss.bz2** contains the logs collected between security audits. This includes information such as doors being opened. ▪ **securityAudit.serial_number.YYYY-MM-DDThhmmss.sss.bz2** contains security audit logs. This includes information such as missing TeraPack magazines or tapes in unexpected slots. The names for both interim security audits and security audits contain the following information: ▪ **serial-number** shows the serial number of the library on which the security audit was run. ▪ **YYYY-MM-DDThhmmss.sss** for an interim security audit indicates the time and date the last audit ended and for a security audit indicates the time and date that the audit began. **YYYY** is the year, **MM** is the two-digit month, **DD** is the two-digit day, **hh** is the 24-hour hour, **mm** is the minute, and **ss.sss** is the seconds and milliseconds.

Command Response The command immediately returns the requested security audit log.

Example Command The following command:

`traces.xml?action=getSecurityAuditLog&name=securityAudit.ZEPPAO1234.2019-03-05T152145.314.bz2` returns the contents of the requested security audit or interim security audit compressed using BZIP2.

The following is the content of an example security audit interim log:

```
[2019-03-04 18:22:36.848]     S95:DiagnosticEvent::SecurityAuditDiagnostic(Finished)
- DiagnosticEvent{type:SecurityAuditDiagnostic, tag:19, state:Finished,
details:Finished running diagnostic test SecurityAuditDiagnostic - diagnostic
details follow:
> ------ Summary ------
> Audited 326 Media in 90 Magazines
>  Unexpected Magazines: 0
>  Missing Magazines: 0
>  Moved Magazines: 0
>
>  Unexpected Media: 0
>  Missing Media: 0
>  Moved Media: 0
> }
[2019-03-05 14:49:12.234]     S96:IncomingAlertEvent::SCM_SAFETY_DOOR_STATE_CHANGED
- IncomingAlertEvent{srcAddr:313, lun:3f, reason:2f, desc:SCM safety door state
changed}
[2019-03-05 14:49:14.234]     S97:IncomingAlertEvent::SCM_SAFETY_DOOR_STATE_CHANGED
- IncomingAlertEvent{srcAddr:313, lun:3f, reason:2f, desc:SCM safety door state
changed}
[2019-03-05 14:52:06.265]     S98:IncomingAlertEvent::SCM_SAFETY_DOOR_STATE_CHANGED
- IncomingAlertEvent{srcAddr:313, lun:3f, reason:2f, desc:SCM safety door state
changed}
[2019-03-05 14:52:08.265]     S99:IncomingAlertEvent::SCM_SAFETY_DOOR_STATE_CHANGED
- IncomingAlertEvent{srcAddr:313, lun:3f, reason:2f, desc:SCM safety door state
changed}
[2019-03-07 17:24:07.233]      E3:EntryExitEvent::import -
EntryExitEvent{media_type:lto, media_barcode:IAN435L5, entry_exit_port:Center TAP}
[2019-03-07 17:24:07.284]      E4:EntryExitEvent::import -
EntryExitEvent{media_type:lto, media_barcode:503404L5, entry_exit_port:Center TAP}
[2019-03-07 17:24:07.338]      E5:EntryExitEvent::import -
EntryExitEvent{media_type:lto, media_barcode:IAN447L5, entry_exit_port:Center TAP}
[2019-03-07 17:38:55.215]    S100:DisplayMessageEven::incorrectTapebarcodeFound -
DisplayMessageEvent(severity:4,messageNum:84,messageString:Quit: not restarting.}
[2019-03-07 18:02:04.921]      E6:EntryExitEvent::export -
EntryExitEvent{media_type:lto, media_barcode:IAN435L5, entry_exit_port:Bulk TAP}
[2019-03-07 18:02:04.976]      E7:EntryExitEvent::export -
EntryExitEvent{media_type:lto, media_barcode:503404L5, entry_exit_port:Bulk TAP}
[2019-03-07 18:02:05.035]      E8:EntryExitEvent::export -
EntryExitEvent{media_type:lto, media_barcode:IAN447L5, entry_exit_port:Bulk TAP}
[2019-03-08 05:14:58.625]    S101:DiagnosticEvent::SecurityAuditDiagnostic(Started)
- DiagnosticEvent{type:SecurityAuditDiagnostic, tag:19, state:Started,
details:Beginning diagnostic test SecurityAuditDiagnostic}
```

The following is the content of an example security audit log:

```
[2019-05-01 13:50:46.849]     S95:DiagnosticEvent::SecurityAuditDiagnostic(Started)
- DiagnosticEvent{tag: 0x19, type: SecurityAuditDiagnostic, state: Started, details:
'Beginning diagnostic test SecurityAuditDiagnostic'}
[2019-05-01 14:01:51.717]     S96:DisplayMessageEvent::magazineRemoved -
DisplayMessageEvent{severity: Warning, type: magazineRemoved, message: 'LUGNG4X
'}
[2019-05-01 14:01:51.755]
S97:DisplayMessageEvent::unknownMagazineAddedToFreePool -
DisplayMessageEvent{severity: Error, type: unknownMagazineAddedToFreePool, message:
'LUDGPBX                          '}
```

```
[2019-05-01 14:01:53.437]
S98:DisplayMessageEvent::verifyPhysicalMagazinesUnexpectedTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesUnexpectedTape,
message: '508672L5        LUDGPBX        '}
[2019-05-01 14:03:24.701]
S99:DisplayMessageEvent::verifyPhysicalMagazinesUnexpectedTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesUnexpectedTape,
message: '518742L5        LUBHB4X        '}
[2019-05-01 14:04:31.527]    S100:IncomingAlertEvent::SCM_SAFETY_DOOR_STATE_CHANGED
- IncomingAlertEvent{srcAddr:313, lun:3f, reason:2f, desc:SCM safety door state
changed}
[2019-05-01 14:05:04.985]
S101:DisplayMessageEvent::magazineMovedContentsUnchanged -
DisplayMessageEvent{severity: Warning, type: magazineMovedContentsUnchanged,
message: 'LU04470                        '}
[2019-05-01 14:06:01.870]
S102:DisplayMessageEvent::verifyPhysicalMagazinesMissingTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesMissingTape,
message: '415263L5        LH000YX        '}
[2019-05-01 14:06:01.904]
S103:DisplayMessageEvent::verifyPhysicalMagazinesMissingTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesMissingTape,
message: '518742L5        LH000YX        '}
[2019-05-01 14:07:07.094]    S104:DisplayMessageEvent::magazineRemoved -
DisplayMessageEvent{severity: Warning, type: magazineRemoved, message: 'LUDGPBX
'}
[2019-05-01 14:07:07.125]
S105:DisplayMessageEvent::verifyPhysicalMagazinesMissingTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesMissingTape,
message: '508672L5        LUDGPBX        '}
[2019-05-01 14:08:17.665]
S106:DisplayMessageEvent::verifyPhysicalMagazinesUnexpectedTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesUnexpectedTape,
message: 'IAN309L5        LUDR6LX        '}
[2019-05-01 14:21:11.249]
S107:DisplayMessageEvent::unknownMagazineAddedToStoragePool -
DisplayMessageEvent{severity: Warning, type: unknownMagazineAddedToStoragePool,
message: 'LU0039X                        '}
[2019-05-01 14:30:49.182]
S108:DisplayMessageEvent::unknownMagazineAddedToFreePool -
DisplayMessageEvent{severity: Error, type: unknownMagazineAddedToFreePool, message:
'LUBGX7X                      '}
[2019-05-01 14:30:50.677]
S109:DisplayMessageEvent::verifyPhysicalMagazinesUnexpectedTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesUnexpectedTape,
message: '418772L5        LUBGX7X        '}
[2019-05-01 14:32:12.795]
S110:DisplayMessageEvent::unknownMagazineAddedToStoragePool -
DisplayMessageEvent{severity: Warning, type: unknownMagazineAddedToStoragePool,
message: 'LUBGOMX                        '}
[2019-05-01 14:32:13.921]
S111:DisplayMessageEvent::verifyPhysicalMagazinesUnexpectedTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesUnexpectedTape,
message: '504273L5        LUBGOMX        '}
```

```
[2019-05-01 14:32:13.948]
S112:DisplayMessageEvent::verifyPhysicalMagazinesUnexpectedTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesUnexpectedTape,
message: 'IAN400L5        LUBGOMX        '}
[2019-05-01 14:43:25.687]    S113:DisplayMessageEvent::magazineRemoved -
DisplayMessageEvent{severity: Warning, type: magazineRemoved, message: 'LUBGX7X
'}
[2019-05-01 14:43:25.692]    S114:IncomingAlertEvent::SCM_SAFETY_DOOR_STATE_CHANGED
- IncomingAlertEvent{srcAddr:313, lun:3f, reason:2f, desc:SCM safety door state
changed}
[2019-05-01 14:43:25.718]
S115:DisplayMessageEvent::verifyPhysicalMagazinesMissingTape -
DisplayMessageEvent{severity: Error, type: verifyPhysicalMagazinesMissingTape,
message: '418772L5        LUBGX7X        '}
[2019-05-01 14:43:25.767]
S116:DisplayMessageEvent::unknownMagazineAddedToStoragePool -
DisplayMessageEvent{severity: Warning, type: unknownMagazineAddedToStoragePool,
message: 'LUGJ66X                        '}
[2019-05-01 14:53:15.478]
S117:DisplayMessageEvent::unknownMagazineAddedToStoragePool -
DisplayMessageEvent{severity: Warning, type: unknownMagazineAddedToStoragePool,
message: 'LUB1U8X                        '}
[2019-05-01 15:10:34.856]   S118:DiagnosticEvent::SecurityAuditDiagnostic(Finished)
- DiagnosticEvent{tag: 0x19, type: SecurityAuditDiagnostic, state: Finished,
details: 'Finished running diagnostic test SecurityAuditDiagnostic - diagnostic
details follow:
> Detected missing magazine with barcode LUGNG4X expected in chamber 2:f:5:10
>   Slots 0:empty, 1:empty, 2:empty, 3:empty, 4:empty, 5:empty, 6:empty, 7:empty,
8:empty, 9:empty
> Detected unexpected magazine with barcode LUDGPBX currently in chamber 2:f:5:10
>   Slots 0:empty, 1:empty, 2:empty, 3:empty, 4:empty, 5:empty, 6:empty, 7:empty,
8:508672L5, 9:empty.
> Detected unexpected tape with barcode 508672L5 in slot 8 in magazine with barcode
LUDGPBX currently in chamber 2:f:5:10.
> Detected unexpected tape with barcode 518742L5 in slot 9 in magazine with barcode
LUBHB4X currently in right robot.
> Detected unexpectedly moved magazine with barcode LU04470 currently in chamber
2:f:5:1; expected in chamber 2:f:5:6
>   Slots 0:empty, 1:508674L5, 2:empty, 3:empty, 4:empty, 5:empty, 6:empty,
7:417314L5, 8:empty, 9:417694L5.
> Detected missing tape with barcode 415263L5 expected in slot 6 in magazine with
barcode LH000YX currently in right robot.
> Detected missing tape with barcode 518742L5 expected in slot 9 in magazine with
barcode LH000YX currently in right robot.

> Detected missing magazine with barcode LUDGPBX expected in chamber 2:f:5:12
>   Slots 0:empty, 1:empty, 2:empty, 3:empty, 4:empty, 5:empty, 6:empty, 7:empty,
8:508672L5, 9:empty
> Detected missing tape with barcode 508672L5 expected in slot 8 in magazine with
barcode LUDGPBX.
> Detected unexpected tape with barcode IAN309L5 in slot 2 in magazine with barcode
LUDR6LX currently in right robot.
> Detected unexpected Inaccessible magazine with barcode LU0039X currently in chamber
2:b:0:17
>   Unverified Slots 0:empty, 1:empty, 2:empty, 3:empty, 4:empty, 5:empty, 6:empty,
7:empty, 8:empty, 9:empty.
```

```
> Detected unexpected magazine with barcode LUBGX7X currently in chamber 3:b:0:18
>   Slots 0:empty, 1:empty, 2:empty, 3:418772L5, 4:empty, 5:empty, 6:empty, 7:empty,
8:empty, 9:empty.
> Detected unexpected tape with barcode 418772L5 in slot 3 in magazine with barcode
LUBGX7X currently in chamber 3:b:0:18.
> Detected unexpected magazine with barcode LUBGOMX currently in chamber 3:f:0:10
>   Slots 0:empty, 1:504273L5, 2:empty, 3:empty, 4:empty, 5:empty, 6:empty, 7:empty,
8:empty, 9:IAN400L5.
> Detected unexpected tape with barcode 504273L5 in slot 1 in magazine with barcode
LUBGOMX currently in chamber 3:f:0:10.
> Detected unexpected tape with barcode IAN400L5 in slot 9 in magazine with barcode
LUBGOMX currently in chamber 3:f:0:10.
> Detected missing magazine with barcode LUBGX7X expected in chamber 3:b:0:19
>   Slots 0:empty, 1:empty, 2:empty, 3:empty, 4:empty, 5:empty, 6:empty, 7:418772L5,
8:empty, 9:empty
> Detected unexpected Inaccessible magazine with barcode LUGJ66X currently in chamber
3:b:0:15
>   Unverified Slots 0:empty, 1:empty, 2:empty, 3:empty, 4:empty, 5:empty, 6:empty,
7:empty, 8:empty, 9:empty.
> Detected missing tape with barcode 418772L5 expected in slot 7 in magazine with
barcode LUBGX7X.
> Detected unexpected magazine with barcode LUB1U8X currently in chamber 3:f:0:15
>   Slots 0:empty, 1:empty, 2:empty, 3:empty, 4:empty, 5:empty, 6:empty, 7:empty,
8:empty, 9:empty.
>
> Attempting to detect moved magazines
>   The missing magazine with barcode LUDGPBX was moved; the contents were found to
be unchanged.  The media pool has been updated; slot numbers may have changed.
>   The missing magazine with barcode LUBGX7X was moved; the contents were found to
be changed!  The media pool has been updated; slot numbers may have changed.
>
> Attempting to detect moved tapes
>   The missing tape with barcode 518742L5 was moved.  The inventory has been
updated.
>   The missing tape with barcode 418772L5 was moved.  The inventory has been
updated.
>
> Did not scan tape with barcode IAN031L5; it is loaded in drive 0:3:0.
> Did not scan Inaccessible magazine with barcode LU0039X currently in chamber
2:b:0:17; it is inaccessible.
> Did not scan Inaccessible magazine with barcode LUGJ66X currently in chamber
3:b:0:15; it is inaccessible.
>


> ------ Summary ------
> Audited 17 tapes in 11 magazines
>   Unexpected magazines: 4
>   Missing magazines:    1
>   Moved magazines:      3
>
>   Unexpected tapes: 3
>   Missing tapes:    1
>   Moved tapes:      2
> '}
>
```

traceType

Description Returns the data for the type of trace specified by the command.

Syntax `traces.xml?traceType=[trace name (see list below)]`

where the value for:

This parameter...	Specifies...
traceType	The name of the trace to be retrieved by the command. See "Capturing Traces" in your library *User Guide* for additional information about traces. Values: ▪ **Action** ▪ **AutoDriveClean** ▪ **AutoSupport** ▪ **BackgroundClient** ▪ **CAN** ▪ **Connection** ▪ **Encryption** ▪ **Error** ▪ **EtherLib** ▪ **Event** ▪ **Geometry** ▪ **GPIO** ▪ **HHM** ▪ **HydraExit** ▪ **Initialization** ▪ **Inventory** ▪ **Kernel** ▪ **Lock** ▪ **LogicalLibrary** ▪ **Message** ▪ **MLM** ▪ **Motion** ▪ **MotionInventory** ▪ **MotionOptions** ▪ **MotionRestart1** ▪ **MotionRestart2** ▪ **PackageUpdate** ▪ **Pools** ▪ **QIP:[*QIP ID*]** ▪ **QIPDump:[*QIP ID*]** ▪ **Security** ▪ **SNMP** ▪ **WebServer** where: ▪ ***QIP ID*** = The component identifier for the (RIM or F-QIP) for which you want to retrieve the specified trace data. The QIP ID is in the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the controller. Not used in T120 library component identifiers. ▪ **F-QIP[*integer*]** = The designator of the controller bay where the QIP is installed. For all libraries except the T120, the value of [***integer***] is always **1**. For the T120 library, the value of [***integer***] is either **1** or **2**.

Command Response The command returns the raw trace data formatted according to the type of trace requested.

Note: The trace data is returned in ASCII format, not XML format.

Example Command and Response The following command:

```
traces.xml?traceType=QIP:FR3/DBA6/F-QIP1
```

returns information about the RIM with the component identifier of FR3/DBA6/F-QIP1.

For example:

```
HW_ISR : initialized
HW_TIM : initialized
startup: PCI bridge parity is ON
startup: PCI bus 1 dev  6 intr  1 funct 0 = 11ab 4620
startup: PCI bus 1 dev  7 intr  0 funct 0 = 1077 2300
startup: PCI bus 0 dev  8 intr  5 funct 0 = 1077 2300
HW_PCI : initialized, found 3 PCI devices
X        26: main   : Module MEM   initialized DRAM
X        31: main   :   MAIN SYS = a03a0000-a07fffff, 00460000h
                                    bytes

...
```

CHAPTER 28

utils

utils.xml

Use the **utils.xml** command to run utilities used for troubleshooting and maintaining the library.

display Barcode Reporting Settings

Description Displays the configurable tape cartridge barcode reporting parameters.

Note: This action was added with BlueScale12.6.45.

Syntax `utils.xml?action=displayBarcodeReportingSettings`

Command Response The command returns the following XML-formatted data:

```
<utils>
   <barcodeReportingSettings>
      <checksummedBehavior>
         nonChecksummedBarcodes|checksummedBarcodes|
         ignoreChecksumBarcodes|unknown
      </checksummedBehavior>
      <directionToStartReportingCharacters>
         left|right|unknown
      </directionToStartReportingCharacters>
      <maxNumberOfCharactersToReport>
         [integer]
      </maxNumberOfCharactersToReport>
   </barcodeReportingSettings>
</utils>
```

where the value for:

This parameter...	Indicates...
checksummed Behavior	Whether the tape barcodes include a checksum character and, if so, how the character is handled. Values: ▪ **nonChecksummedBarcodes** - Labels do not include a checksum. ▪ **checksummedBarcodes** - Labels include a checksum and you want the barcode verified against the checksum when it is read. ▪ **ignoreChecksumBarcodes** - Labels include a checksum character but you do not want the barcode verified against the checksum when it is read. ▪ **unknown** - The checksum type cannot be determined because the response did not match known options.
directionToStart Reporting Characters	Whether the library reports the beginning characters or the end characters on the label if set to not report all characters. Values: ▪ **left** - only the left-most *x* characters in the barcode are reported, where *x* is `maxNumberOfCharactersToReport` ▪ **right** - only the right-most *x* characters in the barcode are reported, where *x* is `maxNumberOfCharactersToReport` ▪ **unknown** - Which characters to report cannot be determined because the response did not match known options.
maxNumberOf CharactersToReport	The maximum number of barcode characters to report. Values: an integer from 1 to 16

Example Command and Response The following command:

```
utils.xml?action=displayBarcodeReportingSettings
```

immediately returns the following XML formatted response:

```
<utils>
   <barcodeReportingSettings>
      <checksummedBehavior>
         ignoreChecksumBarcodes
      </checksummedBehavior>
      <directionToStartReportingCharacters>
         right
      </directionToStartReportingCharacters>
      <maxNumberOfCharactersToReport>
         16
      </maxNumberOfCharactersToReport>
   </barcodeReportingSettings>
</utils>
```

display Robotics Communication Statistics

Description Displays the most recently gathered robotics communication statistics, whether they were gathered using the library user interface or using gatherRoboticsCommunicationStatistics on page 216

Notes:
- This action was added with BlueScale12.8.07.
- This action is only supported on TFinity libraries with high performance transporters. Libraries with legacy transporters return a utility status of **Warning** and a per robot health status of **Indeterminate**.

Syntax `utils.xml?action=displayRoboticsCommunicationStatistics`

Command Response The command returns the following XML-formatted data:

```
<utils>
   <displayRoboticsCommunicationStatistics>
      <status>OK|WARNING|FAILURE</status>
      <message>[message text]</message>
   </displayRoboticsCommunicationStatistics>
</utils>
```

where the value for:

This parameter...	Indicates...
status	The overall status of the utility. Values: ▪ **OK** - The utility ran successfully and all movers report a health status of **HEALTHY**. Or, no robotics communication statistics have been gathered. See gatherRoboticsCommunicationStatistics on page 216. ▪ **WARNING** - The utility ran successfully but at least one mover reports a status of **INDETERMINATE** or **WARNING**. ▪ **FAILURE** - The utility did not run successfully and there is not a report to display.

This parameter...	Indicates...
message	The report from the utility. Each robot reports a COP health status of **Healthy**, **Warning**, or **Indeterminate**. A status of **Indeterminate** indicates that no statistics were available on the specified date, either because there were no moves done or the transporter is not a high performance transporter. **Note:** Most browsers ignore the line feeds in the XML response, making the results difficult to read. The results are easier to read if you run the utility from the BlueScale user interface or run the XML command using a command line utility such as curl.

Example Command and Response The following command:

```
utils.xml?action=displayRoboticsCommunicationStatistics
```

immediately returns the following XML formatted response:

```
<utils>
   <displayRoboticsCommunicationStatistics>
      <status>OK</status>
      <message>
         This library has 2 movers. Threshold for average COP
         transit time is 400 milliseconds (ms). Average COP delay
         time below the threshold is considered healthy. Average
         COP delay transit time equal to greater than the
         threshold value is a warning to contact Spectra Logic
         Technical Support for help. Aggregate Data --------------
         COP health status for Mover 1: HEALTHY. Average transit
         time is 200.307 ms for the mover between 2021-06-14
         12:42:13.850 and 2021-06-15 12:42:13.850 Other values of
         interest for Mover 1 between 2021-06-14 12:42:13.850 and
         2021-06-15 12:42:13.850 are: Transporter Actions: 2781,
         Min Transit Time: 71 ms, Max Transit Time: 7652 ms,
         Standard Deviation: 304.605 ms. COP health status for
         Mover 2: HEALTHY. Average transit time is 212.707 ms for
         the mover between 2021-06-14 12:42:13.850 and 2021-06-15
         12:42:13.850 Other values of interest for Mover 2 between
         2021-06-14 12:42:13.850 and 2021-06-15 12:42:13.850 are:
         Transporter Actions: 3758, Min Transit Time: 77 ms, Max
         Transit Time: 1633 ms, Standard Deviation: 146.261 ms.
         Hourly data for Mover(s) ----------------------- Mover
         1: between 2021-06-14 12:42:13.850 and 2021-06-14
         13:42:13.849 Transporter Actions: 113, Min Transit Time:
         71 ms, Max Transit Time: 1534 ms, Average Transit Time:
         186.009 ms, Standard Deviation: 183.447 ms. Mover 2:
         between 2021-06-14 12:42:13.850 and 2021-06-14
         13:42:13.849 Transporter Actions: 193, Min Transit Time:
         81 ms, Max Transit Time: 784 ms, Average Transit Time:
         196.674 ms, Standard Deviation: 116.859 ms.

         ...
```

```
            Mover 1: - no records for period between 2021-06-15
            10:42:13.850 and 2021-06-15 11:42:13.849 Mover 2: - no
            records for period between 2021-06-15 10:42:13.850 and
            2021-06-15 11:42:13.849 Mover 1: - no records for period
            between 2021-06-15 11:42:13.850 and 2021-06-15
            12:42:13.849 Mover 2: - no records for period between
            2021-06-15 11:42:13.850 and 2021-06-15 12:42:13.849
         </message>
      </displayRoboticsCommunicationStatistics>
   </utils>
```

display TapeBarcode Verification Setting

Description Displays whether the library is configured to perform barcode verification, a scan of the tape barcode, with each robotics move.

Notes:
- This action was added with BlueScale12.6.45.4.
- This action is only supported on TFinity libraries.

Syntax `utils.xml?action=displayTapeBarcodeVerificationSetting`

Command Response The command returns the following XML-formatted data:

```
<utils>
   <tapeBarcodeVerificationSetting>
      <state>on|off</state>
   </tapeBarcodeVerificationSetting>
</utils>
```

where the value for:

This parameter...	Indicates...
state	Whether the tape barcode verification is enabled (on) or disabled (off). Values: **on, off**

Example Command and Response The following command:

```
utils.xml?action=displayTapeBarcodeVerificationSetting
```

immediately returns the following XML formatted response:

```
<utils>
   <barcodeReportingSettings>
   <tapeBarcodeVerificationSetting>
      <state>on</state>
   </tapeBarcodeVerificationSetting>
</utils>
```

gather Robotics Communication Statistics

Description Gathers robotics communication statistics.

Notes:
- This action was added with BlueScale12.8.07.
- This action is only supported on TFinity libraries with high performance transporters. Libraries with legacy transporters return a utility status of **Warning** and a per robot health status of **Indeterminate**.

Syntax `utils.xml?action=gatherRoboticsCommunicationStatistics&date=YYYY-MM-DD[&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
date	The date for which to gather communication statistics in the format *YYYY-MM-DD*. The utility gathers 24 hours of data.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <gatherRoboticsCommunicationStatistics>
      <status>OK</status>
      <message>[message text]</message>
   </gatherRoboticsCommunicationStatistics>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=gatherRoboticsCommunicationStatistics
```

immediately returns the following XML formatted response:

```
<utils>
   <gatherRoboticsCommunicationStatistics>
      <status>OK</status>
      <message>
         gatherRoboticsCommunicationStatistics started. Set
         progress in your query for status.
      </message>
   </gatherRoboticsCommunicationStatistics>
</utils>
```

When the response to `utils.xml?progress` contains `<status>OK</status>`, the command is complete and the the results can be displayed. See displayRoboticsCommunicationStatistics on page 213.

lockTension Rods

Description Changes whether the tension rods are always locked, or whether they can disengage when it is necessary for one TeraPorter in a TFinity library to push the other TeraPorter into its service bay.

Notes:
- This action is only supported on TFinity libraries.
- If the library has BlueScale 12.6.44 or later and permanent locking plates installed or generation 1.5 VAX columns, this utility is ignored.
- Do not change the behavior of the tension rods unless specifically instructed to do so by Spectra Logic Technical Support.
- This action was added with BlueScale12.6.41.

Syntax `utils.xml?action=lockTensionRods&state=[on|off] [&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
state	Whether the tension rods are locked. Values: ▪ **on** = The tension rods are always locked and will not disengage. This is the library's default behavior. ▪ **off** = The tension rods are unlocked and can disengage as necessary.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <lockTensionRods>
      <status>OK</status>
      <message>[message text]</message>
   </lockTensionRods>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=lockTensionRods&state=on
```

immediately returns the following XML-formatted data:

```
<utils>
   <lockTensionRods>
      <status>OK</status>
      <message>lockTensionRods started. Set progress in
         your query for status.</message>
   </lockTensionRods>
</utils>
```

When the response to `utils.xml?progress` contains `<status>OK</status>`, the command is complete and the tension rods are locked.

modify Barcode Reporting Settings

Description Sets the configurable tape cartridge barcode reporting parameters.

Important

- Proceed with caution when changing the library's default barcode reporting settings. Any changes that you make affect every partition in the library. Before making changes, make sure that the changes do not adversely affect others using the library.
- If you modify the barcode reporting settings, you should do so before using any cartridges in the library. If you change the barcode reporting settings after the cartridges have been used by the storage management software, the software may no longer recognize those cartridges. In general, it is best to leave the library set to its defaults and make any reporting modifications on the software side if that is an option.
- Never mix cartridges with checksummed and non-checksummed labels in the library.
- After the command completes, the LCM reboots and any connections to the library through the XML command interface are lost. After the library completes its initialization, it rescans the barcode labels of all media in the library. This rescan takes up to 45 minutes per frame to complete or 20 minutes per frame for a TFinity library with two TeraPorters.

Notes:

- This action was added with BlueScale12.6.45.
- This action is not supported for TFinity libraries running BlueScale12.7.04.03 or later. Instead, see the parameters barcodeLength, barcodeShortened Side, barcodeChecksum, and barcodeChecksum Calculated on page 161 for partition.xml?action=new to set barcode reporting per partition.

Syntax `utils.xml?action=modifyBarcodeReportingSettings&checksummedBehavior=[nonChecksummedBarcodes|checkSummedBarcodes|ignoreChecksumBarcodes]&directionToStartReportingCharacters=[left|right]&maxNumberOfCharactersToReport=[integer][&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
checksummed Behavior	Whether the tape barcodes include a checksum character and, if so, how the character is handled. Values: ▪ **nonChecksummedBarcodes** - Labels do not include a checksum. ▪ **checksummedBarcodes** - Labels include a checksum and you want the barcode verified against the checksum when it is read. ▪ **ignoreChecksumBarcodes** - Labels include a checksum character but you do not want the barcode verified against the checksum when it is read.
directionToStart Reporting Characters	Whether the library reports the left-most or right-most characters on the label if set to not report all characters. Values: ▪ **left** - only the left-most *x* characters in the barcode are reported, where *x* is `maxNumberOfCharactersToReport` ▪ **right** - only the right-most *x* characters in the barcode are reported, where *x* is `maxNumberOfCharactersToReport`
maxNumberOf CharactersToReport	The maximum number of barcode characters to report. Values: an integer from 1 to 16
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <modifyBarcodeReportingSettings>
      <status>OK</status>
      <message>[message text]</message>
   </modifyBarcodeReportingSettings>
</utils>
```

After the command completes, the LCM reboots and any connections to the library through the XML command interface are lost. Wait five to fifteen minutes for the reset to complete and then reconnect.

After the library completes its initialization, it rescans the barcode labels of all media in the library. This rescan takes up to 45 minutes per frame to complete or 20 minutes per frame for a TFinity library with two TeraPorters.

Progress Use `utils.xml?progress` to determine the operation status (see Progress for Extended Action Commands on page 19). The LCM reboots as soon as the command completes.

Example Command and Response The following command:

```
utils.xml?action=modifyBarcodeReportingSettings
&checksummedBehavior= nonChecksummedBarcodes
&directionToStartReportingCharacters=right
&maxNumberOfCharactersToReport=16
```

immediately returns the following XML-formatted data:

```
<utils>
   <modifyBarcodeReportingSettings>
      <status>OK</status>
      <message>modifyBarcodeReportingSettings started. Set
         progress in your query for status.</message>
   </modifyBarcodeReportingSettings>
</utils>
```

When the command completes successfully, the LCM reboots and any connections to the library through the XML command interface are lost. Wait five to fifteen minutes for the reset to complete and then reconnect.

After the library completes its initialization, it rescans the barcode labels of all media in the library. This rescan takes approximately 18 seconds per cartridge.

modify TapeBarcode Verification Setting

Description Modify whether the library is configured to perform tape barcode verification, a scan of the tape barcode, with each robotics move.

Notes:
- This action was added with BlueScale12.6.45.4.
- This action is only supported on TFinity libraries.

Syntax `utils.xml?action=modifyTapeBarcodeVerificationSetting &state=[on|off][&extraInformation=[`*string*`]]`

where the value for:

This parameter...	Specifies...
state	Whether to turn on tape barcode verification. Values: ▪ **on** = Turn on tape barcode verification. Tape barcodes are scanned with each robotics move. ▪ **off** = Turn off tape barcode verification. The tape barcodes are not scanned with robotics moves.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <modifyTapeBarcodeVerificationSetting>
      <status>OK</status>
      <message>[message text]</message>
   </modifyTapeBarcodeVerificationSetting>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=modifyTapeBarcodeVerificationSetting&state=on
```

immediately returns the following XML-formatted data:

```
<utils>
   <modifyTapeBarcodeVerificationSetting>
      <status>OK</status>
      <message>modifyTapeBarcodeVerificationSetting started.
      Set progress in your query for status.</message>
   </modifyTapeBarcodeVerificationSetting>
</utils>
```

When the response to `utils.xml?progress` contains `<status>OK</status>`, the command is complete and tape barcode verification is turned on.

removeAll Library Partitions

Description Removes all partitions configured in the library and then cycles library power.

Important This utility deletes all partitions from the library. Media left in any chamber will be inaccessible when the utility completes.

Note: This action was added with BlueScale12.6.41

Syntax

```
utils.xml?action=removeAllLibraryPartitions
      [&extraInformation=[string]]
```

where the value for:

This parameter...	Specifies...
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <removeAllLibraryPartitions>
      <status>OK</status>
      <message>[message text]</message>
   </removeAllLibraryPartitions>
</utils>
```

When the command completes successfully, library power is cycled and any connections to the library through the XML command interface are lost. Wait five to fifteen minutes for the reset to complete and then reconnect.

Progress Use `utils.xml?progress` to determine the operation status (see Progress for Extended Action Commands on page 19). Library power cycles as soon as the command completes.

Example Command and Response The following command:

```
utils.xml?action=removeAllLibraryPartitions
```

immediately returns the following XML-formatted data:

```
<utils>
   <removeAllLibraryPartitions>
      <status>OK</status>
      <message>removeAllLibraryPartitions started. Set progress in
         your query for status.</message>
   </removeAllLibraryPartitions>
</utils>
```

When the command completes successfully, library power is cycled and any connections to the library through the XML command interface are lost. Wait five to fifteen minutes for the library to initialize and then reconnect.

reset Controller

Description Resets the specified controller (QIP or RIM).

Note: This action was added with BlueScale12.6.45.5.

Syntax `utils.xml?action=resetController&id=[Controller ID]`
`[&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
id	The component identifier for the RIM or F-QIP using the form **FR[*integer*]/DBA[*integer*]/F-QIP[*integer*]**, where: ▪ **FR[*integer*]** = The designator for the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA[*integer*]** = The designator of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **F-QIP[*integer*]** = The designator of the controller bay where the QIP is installed. For all libraries except the T120, the value of [***integer***] is always **1**. For the T120 library, the value of [***integer***] is either **1** or **2**. **Note:** The **ID** values returned by the **controllers.xml ?action=list** command are the component identifiers for the controllers currently installed in the library (see list on page 30).
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <resetController>
      <status>OK</status>
      <message>[message text]</message>
   </resetController>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=resetController&id=FR1/DBA6/F-QIP1
```

immediately returns the following XML-formatted data:

```
<utils>
   <resetController>
      <status>OK</status>
      <message>resetController started. Set progress in
         your query for status.</message>
   </resetController>
</utils>
```

Once the QIP or RIM resets, the response to `utils.xml?progress` contains `<status>OK</status>`.

reset Inventory

Description Reinitializes the cartridge inventory stored by the library. During the reset, the library discards all previous inventory data and rescans all of the magazines and cartridges in the library to establish a new inventory.

Notes:
- Do not run this command unless specifically instructed to do so by Spectra Logic Technical Support.
- The reinitialization process also restarts robotic operations in T200, T380, T680, and T950 libraries.
- This action is not useful for T120 libraries.
- After a Reset Inventory it can take up to 45 minutes per frame to re-inventory a T950 library or 20 minutes per frame to re-inventory a TFinity library with both robots active.

Syntax

`utils.xml?action=resetInventory[&extraInformation=[`*string*`]]`

where the value for:

This parameter...	Specifies...
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <resetInventory>
      <status>OK</status>
      <message>[message text]</message>
   </resetInventory>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=resetInventory
```

immediately returns the following XML-formatted data:

```
<utils>
   <resetInventory>
      <status>OK</status>
      <message>resetInventory started. Set progress in your query
         for status</message>
   </resetInventory>
</utils>
```

Once the inventory resets, the response to `utils.xml?progress` contains `<status>OK</status>`.

resetLCM

Description Resets the LCM, which resets the front panel, web, and XML server for all libraries, and additionally resets robotic operations for T120, T200, T380, and T680 libraries.

Notes:
- This is a forced reset that could leave front panel processes in an unknown state; if you have physical access to the library, use the reset button on the LCM instead.
- This action was added with BlueScale12.6.41.

Syntax `utils.xml?action=resetLCM`

Command Response The command resets the LCM immediately. Any connections to the library through the BlueScale web interface or XML interface are lost. Wait five to fifteen minutes for the reset to complete and then reconnect.

resetRobot

Description Resets the RCM, which restarts all of the control code running on the RCM, including the code that controls the robotics. Any move requests fail until the RCM completes its initialization.

Note: This action is only supported on T950 and TFinity libraries.

Syntax `utils.xml?action=resetRobot[&id=[RCM ID]] [&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
id	The component identifier for the RCM you want to reset, using the form **FR[*integer*]/RCM**, where: ▪ **FR[*integer*]** = The designator for the frame. ▪ **RCM** = Indicates the RCM. **Notes:** ▪ If the **id** parameter is not included, the command resets the RCM in the main frame. ▪ A list of all RCMs in the library can be found in the ECInfo section of the response to the **libraryStatus.xml** command without any parameters. RCM component identifiers are proceeded with "RCM Spectra PC" (see libraryStatus.xml [no parameters] or list on page 73).
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <resetRobot>
      <status>OK</status>
      <message>[message text]</message>
   </resetRobot>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=resetRobot
```

immediately returns the following XML-formatted data:

```
<utils>
   <resetRobot>
      <status>OK</status>
      <message>resetRobot started. Set progress in
         your query for status.</message>
   </resetRobot>
</utils>
```

Once the RCM resets, the response to `utils.xml?progress` contains `<status>OK</status>`.

resetRoboticCalibrationSettings

Description Resets the calibrated state of each drive/slot/chamber so the robotics code recalibrates position the next time the library accesses the drive/slot/chamber.

Caution Remove all tapes from tape drives before running this utility. All loaded tapes are inaccessible after running this utility.

Note: This action is only supported on TFinity libraries.

Syntax `utils.xml?action=resetRoboticCalibrationSettings&robot=both|right|left[&extraInformation=[string]]`

where the value for:

This parameter...	Specifies...
robot	The robot in the library. Left and right are specified as viewed from the front of the library. Values: ▪ **both** = Both robots are reset. ▪ **left** = Reset the left robot as viewed from the front of the library. ▪ **right** = Reset the right robot as viewed from the front of the library.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <resetRoboticCalibrationSettings>
      <status>OK</status>
      <message>[message text]</message>
   </resetRoboticCalibrationSettings>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=resetRoboticCalibrationSettings&robot=both
```

immediately returns the following XML-formatted data:

```
<utils>
   <resetRoboticCalibrationSettings>
      <status>OK</status>
      <message>resetRoboticCalibrationSettings started.
         Set progress in your query for status.</message>
   </resetRoboticCalibrationSettings>
</utils>
```

Once the calibration settings for both robots reset, the response to `utils.xml?progress` contains `<status>OK</status>`.

saveRobotState

Description Saves information about whether a tape is currently in the picker or was in the picker when the robot went into service, and whether a TeraPack magazine is currently in the transporter or was in the transporter when the robot went into service.

Notes:
- This action is only supported on TFinity libraries.
- This action should only be run if a TeraPorter is in a service bay.
- This action was added with BlueScale12.6.45.5.
- Execute libraryStatus refreshEnvironment on page 97 and then libraryStatus [no parameters] or list on page 73 to retrieve the robot state information.

Syntax

```
utils.xml?action=saveRobotState&number=1|2
   [&extraInformation=[string]]
```

where the value for:

This parameter...	Indicates...
number	The number of the robot in the library. Numbering is from left to right as viewed from the front of the library. Values: **1** (left robot) or **2** (right robot). **Note:** To save the state of a robot in a service bay, run the **libraryStatus.xml** command without any parameters (see libraryStatus.xml [no parameters] or list on page 73). Look in the robot section of the response for the robot with a state of `inService` and use the corresponding robot number in this utility.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <saveRobotState>
      <status>OK</status>
      <message>[message text]</message>
   </saveRobotState>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=saveRobotState&number=1
```

immediately returns the following XML-formatted data:

```
<utils>
   <saveRobotState>
      <status>OK</status>
      <message>saveRobotState started. Set progress in
         your query for status.</message>
   </saveRobotState>
</utils>
```

Once the robot state is saved, the response to `utils.xml?progress` contains `<status>OK</status>`.

selective Snowplow

Description Sets the behavior of the transporter when it is putting a TeraPack magazine into a chamber.

Notes:
- Do not enable this option unless specifically instructed to do so by Spectra Logic Technical Support.
- This action is only supported on TFinity libraries.
- This action was added with BlueScale12.6.41.

Syntax `utils.xml?action=selectiveSnowplow&state=[on|off] [&extraInformation=[string]]`

where the value for:

This parameter...	Indicates...
state	Whether the standard transporter behavior for putting a magazine in a chamber is modified. Values: ▪ **on** = The alternate TeraPack magazine behavior is used. ▪ **off** =The standard TeraPack magazine behavior is used.
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <selectiveSnowplow>
      <status>OK</status>
      <message>[message text]</message>
   </selectiveSnowplow>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=selectiveSnowplow&state=on
```

immediately returns the following XML-formatted data:

```
<utils>
   <selectiveSnowplow>
      <status>OK</status>
      <message>selectiveSnowplow started. Set progress in
         your query for status.</message>
   </selectiveSnowplow>
</utils>
```

Once the transporter behavior is set, the response to `utils.xml?progress` contains `<status>OK</status>`.

verify Magazine Barcodes

Description Runs the advanced utility to check all magazine barcodes against the stored inventory. Any moved or added magazine is pulled and its tapes are scanned.

Notes:
- This action is supported on T200, T380, T680, T950, and TFinity libraries.
- This utility only verifies the inventory of tapes within magazines that were moved or added since the last inventory.
- The verification process takes 5 to 10 minutes per frame plus 1 minute for each magazine that was moved or added since the last inventory.
- During the verification process the robot(s) is unavailable.
- This action was added with BlueScale12.6.41.

Syntax
```
utils.xml?action=verifyMagazineBarcodes
    [&extraInformation=[string]]
```

where the value for:

This parameter...	Specifies...
extraInformation (optional)	User specified information to report with the progress action and **taskList.xml** command to help identify background and extended tasks. The string can include any printable character accepted by XML (A-Z, a-z, 0-9, @, -, _, /, ., and space).

Command Response The command returns the following XML-formatted data:

```
<utils>
   <verifyMagazineBarcodes>
      <status>OK</status>
      <message>[message text]</message>
   </verifyMagazineBarcodes>
</utils>
```

Progress Use `utils.xml?progress` to determine the status of the operation. When `<status>` in the command response is **OK** or **FAILED** (see Progress for Extended Action Commands on page 19), it is possible to issue another extended action command.

Example Command and Response The following command:

```
utils.xml?action=verifyMagazineBarcodes
```

immediately returns the following XML-formatted data:

```
<utils>
   <verifyMagazineBarcodes>
      <status>OK</status>
      <message>verifyMagazineBarcodes started. Set progress in
         your query for status.</message>
   </verifyMagazineBarcodes>
</utils>
```

Once the barcode verification completes, the response to `utils.xml?progress` contains `<status>OK</status>`.

INDEX

H

I

K

L

M

O

P

Q

R

S

T

U

V

W

X