SPECTRA
DEEP STORAGE EXPERTS

Spectra Tape Libraries

XML Command Reference

Copyright

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

Part Number

90940114 Revision E

Revision History

Revision	Date	Description
A	June 2012	Initial release.
B	January 2013	Added new actions for driveList.xml.
C	May 2013	Added logout.xml and new action for partition.xml.
D	December 2013	Updated for BlueScale 12.6.3. Added robot status to libraryStatus.xml.
E	April 2015	Added multi drive exporter to partition creation, option keys command, robot service command, new library status information and several new utility commands.

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 http://support.spectralogic.com. The release notes contain updates to the this guide since the last time it was revised.

End User License Agreement

You have acquired a Spectra product that includes software owned or licensed by Spectra Logic from one or more software licensors ("Software Suppliers"). Such software products, as well as associated media, printed materials and "online" or electronic documentation ("SOFTWARE") are protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties.

If you do not agree to this end user license agreement (EULA), do not use the Spectra product; instead, promptly contact Spectra Logic for instructions on return of the Spectra product for a refund. Any use of the Software, including but not limited to use on the Spectra product, will constitute your agreement to this EULA (or ratification of any previous consent).

Grant of License. The Software is licensed on a non-exclusive basis, not sold. This EULA grants you the following rights to the Software:

- You may use the Software only on the Spectra product.
- **Not Fault Tolerant.** The Software is not fault tolerant. Spectra Logic has independently determined how to use the Software in the Spectra product, and suppliers have relied upon Spectra Logic to conduct sufficient testing to determine that the Software is suitable for such use.
- **No Warranties for the SOFTWARE.** The Software is provided "AS IS" and with all faults. The entire risk as to satisfactory quality, performance, accuracy, and effort (including lack of negligence) is with you. Also, there is no warranty against interference with your enjoyment of the Software or against infringement. If you have received any warranties regarding the SOFTWARE, those warranties do not originate from, and are not binding on Software suppliers.
- **Note on Java Support.** The Software may contain support for programs written in Java. Java technology is not fault tolerant and is not designed, manufactured, or intended for use of resale as online control equipment in hazardous environments requiring fail-safe performance, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, direct life support machines, or weapons systems, in which the failure of Java technology could lead directly to death, personal injury, or severe physical or environmental damage.
- **No Liability for Certain Damages.** Except as prohibited by law, Software suppliers shall have no liability for any indirect, special, consequential or incidental damages arising from or in connection with the use or performance of the Software. This limitation shall apply even if any remedy fails of its essential purpose. In no event shall Software suppliers, individually, be liable for any amount in excess of U.S. two hundred fifty dollars (U.S. $250.00).
- **Limitations on Reverse Engineering, Decompilation, and Disassembly.** You may not reverse engineer, decompile, or disassemble the Software, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.
- **Software Transfer Allowed with Restrictions.** You may permanently transfer rights under this EULA only as part of a permanent sale or transfer of the Spectra product, and only if the recipient agrees to this EULA. If the Software is an upgrade, any transfer must also include all prior versions of the Software.
- **Export Restrictions.** Export of the Software from the United States is regulated by the Export Administration Regulations (EAR, 15 CFR 730-744) of the U.S. Commerce Department, Bureau of Export Administration. You agree to comply with the EAR in the export or re-export of the Software: (i) to any country to which the U.S. has embargoed or restricted the export of goods or services, which as May 1999 include, but are not necessarily limited to Cuba, Iran, Iraq, Libya, North Korea, Sudan, Syria, and the Federal Republic of Yugoslavia (including Serbia, but not Montenegro), or to any national or any such country, wherever located, who intends to transit or transport the Software back to such country; (ii) to any person or entity who you know or have reason to know will utilize the Software or portion thereof in the design, development or production of nuclear, chemical, or biological weapons; or (iii) to any person or entity who has been prohibited from participating in U.S. export transactions by any federal agency of the U.S. government. You warrant and represent that neither the BXA nor any other U.S. federal agency has suspended, revoked or denied your export privileges.

Contacting Spectra Logic

To Obtain General Information

Spectra Logic Website: **www.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.
Venture House
Arlington Square, Downshire Way
Bracknell, RG12 1WA
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: **www.spectralogic.com/shop**

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's *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 https://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 *BlueScale Toolbar Option Map* provides a quick reference for locating the options and commands available through the BlueScale user interface.
- 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 TKLM 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 TKLM Server

For additional information that can assist you during the installation and configuration of your server, see the following websites:

- *IBM Tivoli Key Lifecycle Manager Information Center*
- *Tivoli Key Lifecycle Manager Installation and Configuration Guide*

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 and LTO-6)

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: http://www-947.ibm.com/support/entry/portal/Documentation.

TS11*x*0 Technology Drives

The following documents provide information that is applicable to TS11*x*0 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.

Typographical Conventions

This command reference uses the following conventions to highlight important information:

Note: Read text marked with "Note" for additional information or suggestions about the current topic.

⚠ **Important** Read text marked by the "Important" icon for information that will help you complete a procedure or avoid extra steps.

⚠ **Caution** Read text marked by the "Caution" icon for information you must know to avoid damaging the library, the tape drives, or losing data.

⚠ **WARNING** Read text marked by the "Warning" icon for information you must know to avoid personal injury.

CHAPTER 1

Overview

This chapter provides an overview of the XML command interface for the Spectra Tape Series 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. See "Configure Network Settings" in your library's *User Guide* for information and instructions.

Command Categories

The XML commands fall into the following categories and address different aspects of the library's operation:

- **Status and Report**—Commands in this category perform operations such as returning the operational status of the library components, library operating parameters, performance metrics, current configurations, and cartridge inventory. These commands do not modify any system states.
- **Control**—Commands in this category perform operations such as moving tapes from one element in the library to another, performing BlueScale updates, collecting traces, running utilities, and powering off the library.

Issuing Commands

A standard programming language such as Java, Perl, or Python can be used to send a series of XML commands to the library. In addition to sending 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/[command 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
- `[command 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.

Notes:
- In some cases, the base URL can be used without any additional parameters.
- The syntax statements in this guide show these separators in **bold** in 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 (**&**). **Note:** 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.
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.

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's *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 72.
- For full syntax information on the logout command, see logout.xml on page 74.

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. **Notes:** ▪ The specified user must have either superuser or administrative privileges in order to perform configuration operations. ▪ 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. This parameter is optional or can be left blank if no password has been set.

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.

Progress for Asynchronous Commands

When using the BlueScale user interface, certain commands bring the user to a progress page while the command completes asynchronously. Once the command is complete, 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 asynchronous action initiated through the XML command interface, send the base URL command again using the **progress** parameter. The client that issued the original command must request the progress for the command at least one time in order to see the command response.

Important Any time there is an asynchronous action active, the XML command interface is locked out (other than to report progress) until the action completes, regardless of whether the action was initiated from the front panel, through the BlueScale web interface (RLC), or by an XML command.

Notes:

- Commands that are not processed asynchronously return any data associated with the command in the command response without requesting progress.
- To view a list of all asynchronous actions and background operations that the library is processing, use the **taskList.xml** command (see taskList.xml on page 143).

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.

Response (command is in progress)	Response (no command in progress)

`<progress>`
`   <activePage>`
`      [command name]`
`   </activePage>`
`   <message>[message]</message>`
`   <status>`
`      OK|FAILED|QUEUE|SUBMITTED|ACTIVE`
`   </status>`
`</progress>`
	`<progress>`
	`   <message>No Pending Actions</message>`
	`   <status>OK</status>`
	`</progress>`

Example Command The following command:

```
package.xml?progress
```

returns the progress of the package update command.

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.

Using this Command Reference

The following chapters describe each of the available XML commands supported by the Tape Series 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. For example, the **driveList.xml** base URL can be used without parameters, as shown in [no parameters] or list on page 25.
- If the first parameter following the base URL is **action**, the value of the action parameter is used to identify the command variant. For example, the **driveList.xml** base URL can be used with an action parameter as shown in prepareTo ReplaceDrive on page 33.

For each command variant of the base URL, the chapter provides the command syntax, definitions of all variables and parameters, and the syntax error response. 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 Series 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 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 tabs 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

The **autosupport.xml** command is used to generate a new AutoSupport Log (ASL) file or retrieve a previously generated ASL file. ASL files are used during troubleshooting procedures and contain logs of specific library operations.

Command	
generateASL	this page
getASLNames	page 19
getASL	page 20

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

generateASL

Description Generates a new ASL file.

Syntax `autosupport.xml?action=generateASL`

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

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

Progress Use the `autoSupport.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>autosupport.xml</line>
      <line>Query string:</line>
      <line>action=<generateASL|getASLNames|getASL></line>
   </usage>
</syntaxError>
```

Example Command and Response Use the following command to generate a new ASL file and save it:

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

When the response to the `autosupport.xml?progress` indicates that the ASL file generation is complete, use the `autosupport.xml?action=getASLNames` command to display the names of all ASL files, and then use the `autoSupport.xml?action=getASL&name=[ASLName]` command 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) that the ASL file was generated. ▪ *`time`* = The time (hh.mm.ss), based on a 24-hour clock, at which the ASL file was generated.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>autosupport.xml</line>
      <line>Query string:</line>
      <line>action=<generateASL|getASLNames|getASL></line>
   </usage>
</syntaxError>
```

Example Command and Response The following command:

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

retrieves a list of the ASL files currently available on the library.

```
<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 be retrieved by the command.

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

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>autosupport.xml</line>
      <line>Query string:</line>
      <line>action=getASL</line>
      <line>name=[ASL Name]</line>
   </usage>
</syntaxError>
```

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 all of the ASL data.

CHAPTER 3

controllers

controllers.xml

The **controllers.xml** command returns status information for the library controllers (QIPs and RIMs).

Note: Unless otherwise specified, the features of both RIM and RIM2 are the same and “RIM” is used to refer to both.

list

Description Returns controller status, type, firmware, failover configuration, and port configuration information.

Syntax `controllers.xml?action=list`

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

```
<controllers>
   <controller>
      <ID>FRx/DBAx/F-QIPx</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>FRx/DBAx/F-QIPx</failoverFrom>
      <failoverTo>FRx/DBAx/F-QIPx</failoverTo>
      <port>
         <name>A|B</name>
         <useSoftAddress>yes|no</useSoftAddress>
         <loopId>[value]</loopId>
         <initiatorEnabled>yes|no</initiatorEnabled>
         <fibreConnectionMode>
            loop|fabric|automatic
         </fibreConnectionMode>
      </port>
      ...
   </controller>
   ...
<controllers>
```

where the value for:

This parameter...	Indicates...
ID	The component identifier for the exporting RIM or F-QIP using the form **FR*x*/DBA*x*/F-QIP*x*,** where: - **FR*x*** = The number of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. - **DBA*x*** = The number of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. - **F-QIP*x*** = The number of the controller bay where the QIP is installed. For all libraries except the T120, the value of ***x*** is always **1**. For the T120 library, the value of ***x*** is either **1** or **2**.
status	The operating status of the controller. Values: **Normal, Missing,** or **Impaired** (the QIP code fails CAN commands)
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*x*/DBA*x*/F-QIP*x*** (see ID above). Only displays for the secondary controller in a failover pair.
failoverTo	The component identifier of the secondary controller configured in a failover pair, using the form **FR*x*/DBA*x*/F-QIP*x*** (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** - Whether initiator mode in enabled or not. Values: **yes**, **no** - **fibreConnectionMode** - The configured Fibre connection mode. Values: **Loop**, **Fabric**, **Automatic** (auto-negotiated)

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>controllers.xml</line>
      <line>Query string:</line>
      <line>action=<list></line>
   </usage>
</syntaxError>
```

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>Automatic</fibreConnectionMode>
      </port>
      <port>
         <name>A</name>
         <useSoftAddress>yes</useSoftAddress>
         <loopId>0</loopId>
         <initiatorEnabled>no</initiatorEnabled>
         <fibreConnectionMode>Automatic</fibreConnectionMode>
      </port>
   </controller>
</controllers>
```

CHAPTER 4

driveList

driveList.xml

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

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

Command	
[no parameters] or list	page 25
generateDriveTraces	page 30
getDrive Traces	page 31
prepareTo ReplaceDrive	page 33
resetDrive	page 35

[no parameters] or list

Description Returns detailed information about each of the drives 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>
      <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>
      <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>
...
   <drive>
...
   </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 **FRx/DBAx/[*interface*][*technology*]–DRVx**, where: ▪ **FR*x*** = The number of the frame. Only used when the drive is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the drive. Not used with the T120 library. ▪ **[*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**, **SDLT**, **SAIT** ▪ **DRV*x*** = The number 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 24). ▪ Only the T120 library supports SAS drives. ▪ Only the T380, T950, and TFinity libraries support **TS11x0** technology drives. ▪ The **SDLT** and **SAIT** drives are only supported in libraries that contained these drives when purchased. These drives are no longer available for purchase. **EXAMPLE:** In the example command response on page 29, 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’s *User Guide* for additional information about drive component identifiers.
driveStatus	The status of the drive as reported by the library. The status can be: **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. **EXAMPLE:** In the example command response on page 29, the first drive is assigned to a partition named **LTO 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 3592E07 Fibre** = TS11*x*0 technology drives (T380, T950, and TFinity libraries) **EXAMPLE:** In the command response on page 29, the value for the **driveType** parameter of the first drive listed is **IBM Ultrium-TD3 Fibre**.

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.
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. **EXAMPLE:** In the example command response on page 29, the reported serial number for the first drive is **1011000EC2**.
manufacturerSerial Number	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. **EXAMPLE:** In the example command response on page 29, the manufacturer serial number for the first drive is **10380861**.
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. **EXAMPLE:** In the example command response on page 29, the drive firmware version is **93G0**.
dcmFirmware	The firmware version being used by the drive sled that houses the drive. **EXAMPLE:** In the example command response on page 29 the DCM firmware version is **4.7.0**.

This parameter...	Indicates...
wwn	The World Wide Name (WWN) for the drive. See “Drive Connectivity” in your library’s *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**. **EXAMPLE:** In the example command response on page 29, the WWN for the first drive is **21 11 00 90 A5 00 0E C2**.
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’s *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’s *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’s *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’s *User Guide* for detailed information about Drive Lifecycle Management (DLM) and drive health. **Note:** This parameter is only returned if the drive is assigned to a partition. **EXAMPLE:** In the example command response on page 29, the health for the first drive is **Green**.
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.

Syntax Error Response None.

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>
      <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>
      <health>Green</health>
   </drive>
   <drive>
      <ID>FR1/DBA1/fLTO-DRV2</ID>
      <driveStatus>OK</driveStatus>
      <partition>LTO Partition</partition>
      <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>
      <health>Green</health>
   </drive>
</driveList>
```

generateDriveTraces

Description Generate a new drive trace file.

Syntax

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

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 higher drives and TS1140 and higher drives. ▪ *Drive ID* = Is the component identifier for the drive you want to generate a drive trace, using the form **FRx/DBAx/[*interface*][*technology*]–DRVx**, where: ▪ **FR*x*** = The number of the frame. Only used when the drive is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the drive. Not used with the T120 library. ▪ **[*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**, **SDLT**, **SAIT** ▪ **DRV*x*** = The number 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:** ▪ This command can only be used for LTO-5 and later generation drives and TS1140 and later generation 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 24). ▪ See “Drive Identifiers” in your library’s *User Guide* for additional information about drive component identifiers.

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

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

Progress Use the `driveList.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

```
driveList.xml?progress
```

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
      <usage>
         <line>driveList.xml</line>
         <line>Query string:</line>
         <line>action=generateDriveTraces</line>
         <line>driveTracesDrives=[Drive ID]</line>
         <line>driveTracesGetType=<email|download|saveToUSB></line>
         <line>progress(while running)</line>
      </usage>
</syntaxError>
```

Example Command The following command generates a drive trace for drive 1 in DBA1 of frame 1.

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

getDrive Traces

Description Retrieves the last drive trace file generated by the generateDriveTraces action (see generateDriveTraces on page 30).

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	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’s *User Guide* for information about configuring mail recipients.

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>
```

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
      <usage>
         <line>driveList.xml</line>
         <line>Query string:</line>
         <line>action=getDriveTraces</line>
         <line>
            driveTracesGetType=[email|download|saveToUSB]
         </line>
         <line>emailAddress=[Mail Recipient]</line>
      </usage>
</syntaxError>
```

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 of the drive you want to replace using the form **FRx/DBAx/[*interface*][*technology*]–DRVx**, where: ▪ **FR*x*** = The number of the frame. Only used when the drive is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the drive. Not used with the T120 library. ▪ **[*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**, **SDLT**, **SAIT** ▪ **DRV*x*** = The number 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 25). ▪ See “Drive Identifiers” in your library’s *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. ▪ The **SDLT** and **SAIT** drives are no longer available for purchase.

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

```
<prepareToReplaceDrive>
   <status>OK</status>
</prepareToReplaceDrive>
```

Progress Use the `driveList.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>driveList.xml</line>
      <line>Query string:</line>
      <line>action=prepareToReplaceDrive</line>
      <line>driveName=[name of drive to prepare]</line>
      <line>progress(while running)</line>
   </usage>
</syntaxError>
```

Example Command The following command prepares the Fibre Channel drive in drive bay 1 of DBA1 located in frame 1 of a T950 library to be replaced.

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

The following command prepares the Fibre Channel drive in drive bay 1 of DBA3 of a T380 library to be replaced.

```
driveList.xml?action=prepareToReplaceDrive&
   driveName=DBA3/fLTO-DRV1
```

resetDrive

Description Resets the specified drive by power cycling it.

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

where the value for:

This parameter...	Specifies...
driveName	The component identifier of the drive you want to reset using the form **FRx/DBAx/[*interface*][*technology*]–DRVx**, where: ▪ **FR*x*** = The number of the frame. Only used when the drive is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the drive. Not used with the T120 library. ▪ **[*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**, **SDLT**, **SAIT** ▪ **DRV*x*** = The number 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 25). ▪ See "Drive Identifiers" in your library's *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. ▪ The **SDLT** and **SAIT** drives are only supported in libraries that contained these drives when purchased. These drives are no longer available for purchase.

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

```
<resetDrive>
   <status>OK</status>
</resetDrive>
```

Progress Use the `driveList.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15). The drive is then ready to resume operation. If desired, use the **systemMessages.xml** command (see systemMessages.xml on page 140) to retrieve any system messages generated as a result of the reset.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>driveList.xml</line>
      <line>Query string:</line>
      <line>action=resetDrive</line>
      <line>driveName=[name of drive to reset]</line>
      <line>progress(while running)</line>
   </usage>
</syntaxError>
```

Example Command The following command resets drive 1 in DBA1 of frame 1.

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

CHAPTER 5

encryption

encryption.xml

The **encryption.xml** command is used 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 TKLM 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 72) 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>
```

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>encryption.xml</line>
      <line>Query string:</line>
      <line>action=login</line>
      <line>encryptionPassword=[encryption password]</line>
   </usage>
</syntaxError>
```

Example Command The following command:

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

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

CHAPTER 6

HHMData

HHMData.xml

The **HHMData.xml** command retrieves the current status of the Hardware Health Monitoring (HHM) counters for the library and lets you 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’s *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.

[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, which shows the list of HHM counters for the library, the associated threshold values for each counter, and any reminders that have been posted. The data for each counter is returned in a separate counter section of the HHM 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. **EXAMPLE:** In the example command response on page 42, the name of the first counter returned is **Horizontal Axis**.

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). **EXAMPLE:** In the example command response on page 42, the **Life** subType counter for the **Horizontal Axis** has a value of **25172883 inches**; the **Trip1** subType counter has a value of **49513396 inches**.
value	The current value of the counter. **EXAMPLE:** In the example command response on page 42, the **Life** subType counter for the **Horizontal Axis** has a value of **25172883 inches**.
unit	The unit of measure for that data. Values= **inches**, **degrees**, **steps**, **hours**, **moves.** **EXAMPLE:** In the example command response on page 42, the **Trip1** subType counter has a value of **49513396 inches**.
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 46). ▪ 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 43).

This parameter...	Indicates...
severity	The urgency of the reminder. Values = **low**, **medium**, **high**. See "Check and Respond to Messages" in your library's *User Guide* for a description of these system messages. **EXAMPLE:** In the example command response on page 42, a Service HAX reminder was posted for the **Trip1** counter on **2011/08/09 17:08:04**. The reminder has a **low** severity.
defaultThreshold	The value of the factory default setting for the reminder threshold. **EXAMPLE:** In the example command response on page 42, the **Trip1** subType counter for the **Horizontal Axis** has a default threshold value of **35950000.**
currentThreshold	The value of the current reminder threshold (see setThresholdData on page 46 for information about setting the reminder threshold). **EXAMPLE:** In the example command response on page 42, the current threshold value for the **Trip1** subType counter for the **Horizontal Axis** is set to **35950000**.
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:** In the example command response on page 42, a Service HAX reminder was posted for the **Trip1** counter on **2011/08/09 17:08:04.**

Syntax Error Response 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>
            <subType>
               <typeName>Trip1 (Robot 2)</typeName>
...
            </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**. **EXAMPLE:** In the command example on page 45, the HMM counter to reset is **Horizontal Axis**.
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. ▪ Specifying an invalid subType counter causes the command to fail. **EXAMPLE:** In the example command on page 45, the **Trip1** subType counter for the **Horizontal Axis** counter is reset.
robot	**TFinity library only.** For which robot the **Trip1** or **Trip2** counter is being reset. Values: **Robot 1** or **Robot 2**. **Note:** The two robot counters correspond to TeraPorter 1 and TeraPorter 2, respectively.

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>
```

Syntax Error Response

Note: The following syntax error response is for a TFinity library. The response for other libraries does not include the **robot** parameter.

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>HHMData.xml</line>
      <line>Query string:</line>
      <line>action=resetCounterData</line>
      <line>type=[Horizontal Axis|Vertical Axis|
         Picker Axis|Rotational Axis|Magazine Axis|
         Toggle Axis|Side Axis|General Status|Drive to Drive Move|
         Drive to Slot Move|Slot to Slot Move|Slot to Drive Move|
         TAP In Move|TAP Out Move]
      </line>
      <line>subType=[Trip1|Trip2]</line>
      <line>robot=[Robot 1|Robot 2]</line>
   </usage>
</syntaxError>
```

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>
```

setThreshold Data

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 41) 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 41 for additional information. **EXAMPLE:** In the example command on page 48, the reminder threshold for the Service HAX Belt reminder is set to **46300000**. When a Trip1 or Trip2 counter that uses this reminder reaches this value, the library will add a reminder to the data returned by the **HHMData.xml** command. The command response indicates that the threshold was set successfully.

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>
```

Syntax Error Response

Note: The following syntax error response is for a TFinity library. The syntax error response from other libraries will list different events.

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>HHMData.xml</line>
      <line>Query string:</line>
      <line>action=setThresholdData</line>
      <line>event=[Service HAX|Check Contact Brushes|Service
         VAX|Service VAX Belt|Service Transporter|Service
         Required]
      </line>
      <line>default=[true|false](must provide either default=true
         or a value)
      </line>
      <line>
         value=[value to set the threshold (1 thru 4294967295)]
      </line>
   </usage>
</syntaxError>
```

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 7

inventory

inventory.xml

The **inventory.xml** command retrieves information about the inventory status of all the slots and drives assigned to a specified partition.

partition

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?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 126). ▪ The partition name is set when the partition is created. See partition.xml on page 101 for information about using the XML command interface to configure partitions in the library. Refer to your library's *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>
            <barcode>[value]</barcode>
            <isQueued>[value]</isQueued>
            <full>yes|no</full>
         </drive>
   </partition>
</inventory>
```

where the value for:

This parameter...	Indicates...
name	The name of the partition. **EXAMPLE:** In the command response on page 53 the partition name is **Partition 1**.
storageSlot	That the following block of information is for a slot in the partition storage pool. **EXAMPLE:** In the command response on page 53, the first block of information following the **storageSlot** parameter contains the data for the cartridge slots assigned to the Partition 1 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. **EXAMPLE:** In the command response on page 53, the block of information following the **entryExitSlot** section contains the data for the cartridge slots assigned to the Partition 1 entry/exit pool.
drive	That the following block of information is for a drive assigned to the partition. **EXAMPLE:** In the command response on page 53, the last block of information contains the data for the drives assigned to Partition 1.
id	The SCSI element address of the slot or drive that the library reports to the host. **EXAMPLE:** In the example command response on page 53, the element address for the first storage slot listed is **4096**.
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's *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's *User Guide* for detailed information about creating a move queue. **EXAMPLE:** In the example command response on page 53, the **offset** value for the first slot listed for the storage pool is **1**. The empty slot in the storage pool has an **offset** value of **47**. The second drive in the partition has an **offset** value of **2**.

This parameter...	Indicates...
barcode	The cartridge 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**. **EXAMPLE:** In the example command response on page 53 the barcode for the cartridge in the first storage slot listed is **83270L4**.
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**.
full	Whether the slot or drive is full. Values: **yes**, **no**

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?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>031795L4</barcode>
         <isQueued>no</isQueued>
         <full>yes</full>
      </drive>
      <drive>
         <id>257</id>
         <offset>2</offset>
         <full>no</full>
      </drive>
   </partition>
</inventory>
```

CHAPTER 8

libraryStatus

libraryStatus.xml

The **libraryStatus.xml** command returns status information for the library.

Topic	
[no parameters]	below
refreshECInfo	page 70
refresh Environment	page 71

[no parameters]

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>
   <railPowerOn>yes|no</railPowerOn>
   <robot>
      <number>1|2</number>
      <state>inService|impaired|good|unknown</state>
      <serviceFrame>left|right|none|unknown</serviceFrame>
      <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>
   </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>
      <tempparatureAlarm>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>

</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**
railPowerOn TFinity libraries only	Whether the rail power is on (yes) or off (no). Values: **yes, no**
robot TFinity libraries only	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** ▪ **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**
serialNumber	The library serial number (hardware ID).

This parameter...	Indicates...
exsessiveMove 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 most recent Sense Key, Additional Sense Code (ASC), Additional Sense Code Qualifiers (ASQC). ▪ **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 using the form **FR*x*/DBA*x*/F-QIP*x*,** where: ▪ **FR*x*** = The number of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **F-QIP*x*** = The number of the controller bay where the QIP is installed. For all libraries except the T120, the value of ***x*** is always **1**. For the T120 library, the value of ***x*** 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**

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*x*/DBA*x*/DRV*x***, where: ▪ **FR*x*** = The number of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **DRV*x*** = The number of the drive bay in the DBA. 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*x*/PCM/PowerSupply*x*** where: ▪ **FR*x*** = 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*x*** = 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** - 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 **FRx/PCM** where: ▪ **FR*x*** = The number 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 will always show 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 will always show 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** ▪ **faulted** = Whether the power supply is faulted. Values: **yes, no**

This parameter...	Indicates...
controller EnvironmentInfo (*continued*)	**fanControlModule** - 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 the frame. This is only applicable to T200, T280, T680, and T950 (not including T950B) libraries. ▪ **ID** - The component identifier for the module using the form **FR*x*/FCM** where: ▪ **FR*x*** = The number 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 will always show a value of **no.** ▪ **newFilterCalibrated** = This is no longer used. It will always show 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 T950 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*x*/FMM** where: ▪ **FR*x*** = The number 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 switch 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.
controller EnvironmentInfo (*continued*)	**serviceBayControlModule** - The service bay control module, located in each TFinity service frame, controls the service frame interlocks and provides bulk TAP status. This is only applicable for TFinity libraries. ▪ **ID** - The component identifier for the module using the form **FR*x*/SCM** where: ▪ **FR*x*** = The number 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: **closed**, **open**. ▪ **sideAccessPanel** = The state of the side access panel interlock switch. Values: **closed**, **open**. ▪ **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.

Syntax Error Response None

Example Command and Response The following command:

```
libraryStatus.xml
```

immediately returns the following XML-formatted data when issued to a T950 library:

```
<libraryStatus>
   <libraryType>T950</libraryType>
   <serialNumber>1137402</serialNumber>
   <excessiveMoveFailures>
      <move>
         <partition>Partition 1</partition>
         <source>Slot 28</source>
         <destination>Slot 361</destination>
         <numberOfFailures>5</numberOfFailures>
         <lastSenseInfo>0x04, 0x85, 0x02</lastSenseInfo>
         <lastFailedMoveTime>
            2015/01/07:18:04:27
         </lastFailedMoveTime>
      </move>
      ...
   </excessiveMoveFailures>
   <controllerEnvironmentInfo>
      <controller>
         <ID>FR1/DBA6/F-QIP1</ID>
         <temperatureInCelsius>33</temperatureInCelsius>
         <portALoopUp>yes</portALoopUp>
         <portBLoopUp>yes</portBLoopUp>
      </controller>
```

```
<driveControlModule>
   <ID>FR1/DBA6/fTS11x0-DRV1</ID>
   <twelveVoltVoltage>12.000</twelveVoltVoltage>
   <fiveVoltVoltage>5.011</fiveVoltVoltage>
   <fanCurrentInAmps>0.142</fanCurrentInAmps>
   <temperatureInCelsius>27</temperatureInCelsius>
</driveControlModule>
...
<powerSupplyFRU>
   <ID>FR1/PCM/PowerSupply1</ID>
   <inputPowerOkay>yes</inputPowerOkay>
   <outputPowerOkay>yes</outputPowerOkay>
   <temperatureWarning>no</temperatureWarning>
   <temperatureAlarm>no</temperatureAlarm>
   <modelNumber>SP525-Y01A</modelNumber>
   <manufacturerPartNumber>SP525-Y01A</manufacturerPartNumber>
   <serialNumber>TE30546</serialNumber>
   <modLevel>A</modLevel>
   <manufacturer>CHEROKEE INTL</manufacturer>
   <countryOfManufacturer>USA</countryOfManufacturer>
   <temperatureInCelsius>29</temperatureInCelsius>
   <communicatingWithPCM>yes</communicatingWithPCM>

   <fanInPowerSupplyFRU>
      <number>1</number>
      <okay>yes</okay>
   </fanInPowerSupplyFRU>
   ...

   <powerSupplyInPowerSupplyFRU>
      <nominalVoltage>12.000</nominalVoltage>
      <actualVoltage>12.069</actualVoltage>
      <actualCurrentInAmps>4.976</actualCurrentInAmps>
   </powerSupplyInPowerSupplyFRU>

   ...

</powerSupplyFRU>

...
```

```
<powerControlModule>
   <ID>FR1/PCM</ID>
   <parallelACPresent>no</parallelACPresent>
   <primaryACPresent>no</primaryACPresent>
   <secondaryACPresent>no</secondaryACPresent>
   <supplyDetectionWorking>yes</supplyDetectionWorking>
   <ACCurrentInAmps>0</ACCurrentInAmps>
   <primaryACVoltage>19</primaryACVoltage>
   <secondaryACVoltage>53</secondaryACVoltage>
   <twelveVoltVoltage>11.901</twelveVoltVoltage>
   <fiveVoltVoltage>4.931</fiveVoltVoltage>
   <onBoardTemperatureInCelsius>
      25
   </onBoardTemperatureInCelsius>
   <remoteTemperatureInCelsius>
      0
   </remoteTemperatureInCelsius>
   <powerSupplyInPCM>
      <position>1</position>
      <faulted>no</faulted>
   </powerSupplyInPCM>
   <powerSupplyInPCM>
      <position>9</position>
      <faulted>no</faulted>
   </powerSupplyInPCM>
</powerControlModule>
<fanControlModule>
   <ID>FCM</ID>
   <frameNumber>1</frameNumber>
   <temperatureInCelsius>22</temperatureInCelsius>
   <backPanelSwitch>closed</backPanelSwitch>
   <fanPanelSwitch>closed</fanPanelSwitch>
   <filterPanelSwitch>closed</filterPanelSwitch>
   <boardVoltage>24.024</boardVoltage>
   <fanInputVoltage>12.000</fanInputVoltage>
   <fanSpeedVoltage>9.006</fanSpeedVoltage>
   <fanSpeedSetting>2</fanSpeedSetting>
   <newFansCalibrated>no</newFansCalibrated>
   <newFilterCalibrated>no</newFilterCalibrated>
   <fanInFCM>
      <number>1</number>
      <poweredOn>yes</poweredOn>
      <speedInRPM>4320</speedInRPM>
   </fanInFCM>
   ...
   <lightBank>
      <number>1</number>
      <poweredOn>yes</poweredOn>
   </lightBank>
   ...
</fanControlModule>
```

```
      <frameManagementModule>
         <ID>FR1/FMM</ID>
         <frameIDInfo>0x80</frameIDInfo>
         <twentyFourVoltVoltage>23.914</twentyFourVoltVoltage>
         <fiveVoltVoltage>5.136</fiveVoltVoltage>
         <fanRailVoltage>13.847</fanRailVoltage>
         <switchedRailVoltage>23.914</switchedRailVoltage>
         <twentyFourVoltCurrentInAmps>4.525
            </twentyFourVoltCurrentInAmps>
         <powerConsumedInWatts>0</powerConsumedInWatts>
         <sampleRateInSeconds>2</sampleRateInSeconds>
         <samplesTaken>52</samplesTaken>
         <temperatureInCelsius>24</temperatureInCelsius>
         <EPMTemperatureInCelsius>44</EPMTemperatureInCelsius>
         <frameToFrameTemperatureInCelsius>44
            </frameToFrameTemperatureInCelsius>
         <frameToFrameAttached>yes</frameToFrameAttached>
         <frameToFrame5VoltEnabled>yes</frameToFrame5VoltEnabled>
         <fansEnabled>yes</fansEnabled>
         <backSwitchOpen>no</backSwitchOpen>
         <filterSwitchOpen>no</filterSwitchOpen>
         <frontSwitchOpen>no</frontSwitchOpen>
         <safetyInterlockOpen>no</safetyInterlockOpen>
         <driveFrameNumber>0x7</driveFrameNumber>
         <switchedRailState>24Volt</switchedRailState>
         <robotPowerEnabled>yes</robotPowerEnabled>
         <internalLightsEnabled>yes</internalLightsEnabled>
         <externalLightsEnabled>yes</externalLightsEnabled>

         <fanPair>
            <number>1</number>
            <present>yes</present>
         </fanPair>
         ...

         <fan>
            <number>1</number>
            <speedInRPM>8280</speedInRPM>
         </fan>
         ...
   </controllerEnvironmentInfo>
   <ECInfo>
      <component><ID>LCM Spectra PC</ID>
         <EC>5</EC>
         <serialNumber>LS31133012</serialNumber>
         <topLevelAssemblyEC>2</topLevelAssemblyEC>
         <topLevelAssemblySerialNumber/>
         <date>08/22/2013</date>
      </component>
      ...

   </ECInfo>

</libraryStatus>
```

refreshECInfo

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

Syntax `libraryStatus.xml?action=refreshECInfo`

Command Response The command 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>
```

Progress Use the `libraryStatus.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>libraryStatus.xml</line>
      <line>Query string:</line>
      <line>action=[refreshECInfo|refreshEnvironment]</line>
      <line>progress (while running)</line>
   </usage>
</syntaxError>
```

refresh Environment

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

Syntax `libraryStatus.xml?action=refreshEnvironment`

Command Response The command 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>
```

Progress Use the `libraryStatus.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>libraryStatus.xml</line>
      <line>Query string:</line>
      <line>action=[refreshECInfo|refreshEnvironment]</line>
      <line>progress (while running)</line>
   </usage>
</syntaxError>
```

CHAPTER 9

login

login.xml

The **login.xml** command is used 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's *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's *User Guide*) or can be closed by issuing a logout command, see logout.xml on page 74.

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 administrative privileges in order to perform configuration operations. ▪ 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.
password (optional)	The password associated with the username. **Notes:** ▪ If no password is set, this parameter can be left blank or not included. ▪ The password is case sensitive.

This parameter...	Specifies...
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 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.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>login.xml</line>
      <line>Query string:</line>
      <line>username=[valid user name]</line>
      <line>password=[valid password]</line>
      <line>forceFrontPanel (optional)</line>
   </usage>
</syntaxError>
```

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 10

logout

logout.xml

The **logout.xml** command is used 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's *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.

Syntax `logout.xml`

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

```
<logout/>
```

Syntax Error Response None.

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 72).

CHAPTER 11

mediaExchange

mediaExchange.xml

The **mediaExchange.xml** command is used 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.

Topic	
clean	this page
getTAPState	page 77
importExport	page 80
Importing, Exporting, or Exchanging Cartridges	page 84

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 command 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 80 to import the magazines into the correct partitions.
- The clean command 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]
```

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 126). **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. **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 not supported by libraries that do not include a bulk TAP. In addition, the command fails if the library does not have a bulk TAP installed on the side specified by the parameter value.

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

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

Progress Use the following command to determine when the clean operation is complete (see Progress for Asynchronous Commands on page 15).

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

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. **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 not supported by libraries that do not include a bulk TAP. In addition, the command fails if the library does not have a bulk TAP installed on the side specified by the parameter value.

When the command response is **No Pending Actions**, the clean operation for the specified bulk TAP is complete.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>mediaExchange.xml</line>
      <line>Query string:</line>
      <line>action=clean</line>
      <line>partition=[partition name]</line>
      <line>TAPDevice=[main|leftBulk|rightBulk]</line>
   </usage>
</syntaxError>
```

Example Command and Response The following command:

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

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. Issue a `mediaExchange.xml?progress&TAPDevice=leftBulk` command to determine when the clean operation is complete.

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 not supported by libraries that do not include a bulk TAP. In addition, 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|SDLT|SAIT|TS11x0|
      TS11x0 Maintenance|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.

This parameter...	Indicates...
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**, **SDLT**, **SAIT**, **TS11x0**, **TS11x0 Maintenance**, **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 **SDLT** and **SAIT** drives are no longer available for purchase. ▪ The **TS11x0** and **TS11x0 Maintenance** values are only supported in T950, and TFinity libraries that have TS11*x*0 technology drives installed.
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**.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>mediaExchange.xml</line>
      <line>Query string:</line>
      <line>action=getTAPState</line>
      <line>TAP=[mainTop|mainBottom|leftBulk|rightBulk]</line>
      <line>drawerNumber=[1...14](optional, default=1)</line>
   </usage>
</syntaxError>
```

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 **phyInventory.xml** command to determine the status of each inventory location in the partition (see physInventory.xml on page 127).

- 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 75). 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 will fail. 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 75).

- 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 cartridges, remove the magazines from the carousel one at a time, make the desired cartridge exchanges, and then put the magazine back into the same location that it originally occupied.

⚠ **Important** 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]&
    timeoutInMinutes=[value]&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 the partitions currently configured in the library (see partitionList.xml on page 126). **Notes:** ▪ The partition name is case-sensitive. ▪ The partition name is set when the partition is created.

This parameter...	Specifies...
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’s *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. **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 not supported by libraries that do not include a bulk TAP. In addition, the command fails if the library does not have a bulk TAP installed on the side specified by the parameter value.
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 127). **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.

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

```
<mediaExchange>
   <message>
      Media export/import started for main|leftBulk|rightBulk TAP
   </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 If desired, you can use the following command to monitor the progress of the importExport command (see Progress for Asynchronous Commands on page 15).

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

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. **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 not supported by libraries that do not include a bulk TAP. In addition, the command fails if the library does not have a bulk TAP installed on the side specified by the parameter value.

When the command response is **No Pending Actions**, the command is complete.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>mediaExchange.xml</line>
      <line>Query string:</line>
      <line>action=importExport</line>
      <line>partition=[partition name]</line>
      <line>slotType=[storage|IE]</line>
      <line>
         timeoutInMinutes=[1...10080](optional, default=10)
      </line>
      <line>TAPDevice=[main|leftBulk|rightBulk]</line>
      <line>TeraPackOffsets[n,n,n...]</line>
   </usage>
</syntaxError>
```

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 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 will 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's *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 127 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 command is complete.

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

Repeat the progress command as often as desired until the response is **No Pending actions**.

Import Magazines Example

1. After performing the steps in Preparation on page 84, 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 130.

   ```
   mediaExchange.xml?action=importExport&partition=Partition 1&
      slotType=IE&TAPDevice=leftBulk&
      TeraPackOffsets=61,62,63,64,65,66
   ```

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.

Important 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, remove 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 is **No Pending actions**.

Export or Exchange Magazines or Cartridges Example

1. After performing the steps in Preparation on page 84, 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 130.

   ```
   mediaExchange.xml?action=importExport&partition=Partition 1&
      slotType=storage&TAPDevice=leftBulk&TeraPackOffsets=1,6
   ```

   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 or make the desired cartridge exchanges.

Important 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, 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 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 in complete.

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

   Repeat the progress command as often as desired until the response is **No Pending actions**.

CHAPTER 12

optionKeys

optionKeys.xml

The **optionKeys.xml** command adds a new option (activation) key to the library and lists 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.

Topic	
add	page 87
list	page 87

add

Description Adds a new activation key to the library.

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

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>
```

Syntax Error Response

```
<errorMessage>
   <message>[error message text]</message>
   <usage>
      <line>optionkeys.xml</line>
      <line>Query string:</line>
      <line>action=add</line>
      <line>key=[key]</line>
   </usage>
</errorMessage>
```

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.

Syntax Error Response None

Example Command and Response The following command:

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

retrieves a list of activation keys entered into the library.

```
<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 13

package

package.xml

The **package.xml** command is used to update the library using a BlueScale package previously uploaded to the memory card in the LCM using the **packageUpload.xml** command (see packageUpload.xml on page 99) and to check the results of the update operation.

You can also use the **package.xml** command to retrieve the name of the BlueScale package currently being 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, including the one currently in use.

Topic	
[no parameters] or list	this page
getResults	page 91
update	page 93
Updating the Library BlueScale Software	page 97

[no parameters] or list

Description Retrieves the name of the BlueScale package currently being 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, including the one currently in use.

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 full version package 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**. The BlueScale user interface also displays **None** for the BlueScale version shown on the status bar and in the output of the Get Firmware Versions utility. **EXAMPLE:** In the command example below, the full name of the current BlueScale version is **BlueScale12.0.3-20111122F**.
list	The version names of each BlueScale package currently stored on the library. **Note:** If there are no package files stored in the library, the **list** section is not returned. **EXAMPLE:** In the command example below, the following BlueScale versions are currently stored on the library memory card: **BlueScale12.0.3–20111122F** and **BlueScale11.3.3-20110316F**.

Syntax Error Response None.

Example Command The following command:

```
package.xml
```

returns the current BlueScale package information:

```
<package>
   <current>
      <name>BlueScale12.0.3–20111122F</name>
   </current>
   <list>
      <name>BlueScale12.0.3–20111122F</name>
      <name>BlueScale11.3.3-20110316F</name>
   </list>
</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 command, you must send a `package.xml?action=getResults` command in order to complete the update process. See getResults on page 91 for more information.

Notes:

- Running this command 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 on autoFinish on page 95.
- 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 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>
      <component>
...
      </component>
      <rebootInProcess>true|false</rebootInProcess>
   </updateResults>
</package>
```

where the value for:

This parameter...	Indicates...
packageName	The full name of the BlueScale package that was used to update the library. **EXAMPLE:** In the command response on page 93, the package name is **BlueScale12.1.0-20120423F**.
name	The name of the BlueScale component that was updated. **EXAMPLE:** In the command response on page 93, the name of the first component that was updated is **LC OS**.
previousVersion	The version number of the BlueScale component before the update. **EXAMPLE:** In the command response on page 93, the version number for the LC OS component before the update was **100.4.0.0**.
updatedVersion	The version number of the BlueScale component following the update. **EXAMPLE:** In the command response on page 93, the new version number for the LC OS component is **100.5.0.0**.
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. **EXAMPLE:** In the command response on page 93, the status of the update for the LC OS component is **OK**.
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. **EXAMPLE:** In the command response on page 93, the value for the **rebootInProcess** parameter is **false**, indicating that the update did not require a reboot of the LCM and RCM.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>package.xml</line>
      <line>Query string:</line>
      <line>action=[update|getResults]</line>
   </usage>
</syntaxError>
```

Example Command and Response The following command:

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

returns the following information for the example shown on page 96.

```
<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>
      <component>
...
      </component>
      <rebootInProcess>false</rebootInProcess>
   </updateResults>
</package>
```

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 already present, you must first upload the necessary BlueScale package (see packageUpload.xml on page 99).

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’s *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 will fail if the library does not have a valid BlueScale Software Support key installed.

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

- Stop all library operations.

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

- All backup 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.

- 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's *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's *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`

where the value for:

This parameter...	Specifies...
package	The version 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 will reboot the LCM and RCM when the update is complete. 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.

Update Process After the library receives and validates the command, it begins applying the updates in the specified package file. The update command returns a status indicating that the update has started.

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

Command Response The command returns the following XML-formatted data indicating whether the package command ran successfully:

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

Progress Use the `package.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

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 `update.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 91 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 91 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 140).
- If the command included the **autoFinish** parameter and the update require a reboot, you can retrieve the update results using the `traces.xml?traceType=Message` command (see traces.xml on page 145).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>package.xml</line>
      <line>Query string:</line>
      <line>action=update</line>
      <line>package=[package name]</line>
      <line>autoFinish (optional)</line>
   </usage>
</syntaxError>
```

Example Command The following command:

```
package.xml?action=update&package=BlueScale12.1.0-20120423F&
   autoFinish
```

updates the library to BlueScale12.1.0-20120423F. When the update is complete, the LCM or RCM may reboot 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 BlueScale" in your library's *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 99).
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 will be 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 a **No Pending actions** 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 140 and traces.xml on page 145 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 uploaded should be listed in the **current** section of the XML data returned in the command response.

CHAPTER 14

packageUpload

packageUpload.xml

The **packageUpload.xml** command is used within a script to upload a BlueScale package file to the library's memory card. After the package file is uploaded, it can be used to update the BlueScale software that the library is running (see package.xml on page 89).

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.

[no parameters]

The **packageUpload.xml** command is used to upload 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.

Important Using Python requires the MultipartPostHandler software library, which can be downloaded free of charge from:
pipe.scs.fsu.edu/PostHandler/MultipartPostHandler.py

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

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

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>packageupload.xml</line>
      <line>POST data encoded in the following way:</line>
      <line>--[user defined boundary]--</line>
      <line>Content-Disposition: form-data; name="BlueScalePkg";
         filename="[packageFile]"
      </line>
      <line>Content-Type: application/octet-stream</line>
      <line></line>
      <line>[packageFile Data];</line>
      <line></line>
      <line>--[user defined boundary]--</line>
   </usage>
</syntaxError>
```

CHAPTER 15

partition

partition.xml

The **partition.xml** command is used to configure one or more data or cleaning partitions in the library. See the “Configuring and Managing Partitions” chapter in your library’s *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 126).

Topic	
autoCreate	page 101
delete	page 104
list	page 106
new	page 112

autoCreate

Description Automatically creates a single storage partition that uses all the chambers (or slots in the T120 library) that are enabled by the CoD key and all installed drives.

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 will fail.
- 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, you will need to 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, you will need to 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 need 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]]
```

where the value for:

This parameter...	Specifies...
partition	The name you want to use for the partition. **Note:** The partition name is case-sensitive.
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’s *User Guide* for information about configuring mail users.

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

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

Progress Use the `partition.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>partition.xml</line>
      <line>Query string:</line>
      <line>action=autoCreate</line>
      <line>partition=[partition name]</line>
      <line>
         saveLibraryConfiguration=[USB|emailRecipient](optional)
      </line>
      <line>progress (while running)</line>
   </usage>
</syntaxError>
```

Example Command and Response The following command:

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

creates a single storage partition named **Partition 1** and emails the configuration backup file to the mail user **JaneSuperuser@YourCompany.com**.

delete

Description Deletes the specified partition from the library. When you delete a partition, the drives and chambers previously assigned to that partition can be reassigned to an existing partition or used to create 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 75 or “Exporting or Exchanging Magazines and Cartridges” in your library’s *User Guide*.

Important After the partition is deleted, any magazines in the chambers that were assigned to the partition’s storage and entry/exit pools are not accessible until the chambers are assigned to another partition.

- If the storage partition is configured to use encryption, make sure that you have exported the encryption key(s) for any cartridges that were in the partition as described in the *Spectra Encryption User Guide*. The encryption key(s) will be required if you import the cartridges into another partition in order to access the data on the cartridges.
- 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]]`

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 126).
saveLibrary Configuration	Where you want to save the configuration backup file that the library generates after the partition is deleted. 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:** ▪ 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, make sure that the USB device is connected to the LCM before running the command. ▪ See “Configure Mail Users” in you library’s *User Guide* for information about configuring mail users.

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

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

Progress Use the `partition.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>partition.xml</line>
      <line>Query string:</line>
      <line>action=delete</line>
      <line>parition=[paritition name]</line>
      <line>
         saveLibraryConfiguration=[USB|emailRecipient](optional)
      </line>
      <line>progress (while running)</line>
   </usage>
</syntaxError>
```

Example Command and Response The following command:

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

deletes the partition named **Partition 1** and emails the configuration backup file to the mail user **JaneSuperuser@YourCompany.com**.

list

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

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|Super AIT|
         Super DLT
      </type>
      <emulation>[emulation type]</emulation>
      <includeDriveAndMediaGenerationInRES>
         yes|no
      </includeDriveAndMediaGenerationInRES>
      <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|SpectraTKLM
         </type>
         <encryptionKey>[moniker]</encryptionKey>
         <decryptionKeys>
            <moniker>[moniker]</moniker>
            …
         </decryptionKeys>
      </encryption>
      <moveOptionType>default|virtual</moveOptionType>
   </partitionData>
   …
</partition>
```

where the value for:

This parameter...	Specifies...
partitionData	The beginning of data for one partition.

This parameter...	Specifies...
name	The name of the partition. The partition name is case-sensitive.
type	The type of partition. Values: **LTO**, **LTO Cleaning**, **TS11x0**, **TS11x0 Cleaning**, **Super DLT**, **Super AIT** **Note:** The only partitionData which 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”
includeDriveAndMediaGenerationInRES	Whether the response to the SCSI Read Element Status command will include drive and media generation data. Values: **yes, no**
exporters	The name, type, and addressing for the controller (RIM, F-QIP) or direct-attach drive(s) that provides 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*x*/DBA*x*/F-QIP*x***, where: ▪ **FR*x*** = The number of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **F-QIP*x*** = The number of the controller bay where the QIP is installed. For all libraries except the T120, the value of ***x*** is always **1**. For the T120 library, the value of ***x*** is either **1** or **2**. ▪ For a direct-attached drive, **ID** = The component identifier for the exporting drive(s) using the form **FR*x*/DBA*x*/DRV*x***, where: ▪ **FR*x*** = The number of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **DRV*x*** = The number 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. ▪ **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.
slotsPerChamber	The number of slots per magazine. This number depends on the technology. Each LTO magazine has slots for 10 LTO cartridges; a TS11*x*0 technology 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*x*0 technology partition, the number of slots must be a multiple of 9.

This parameter...	Specifies...
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*x*0 technology 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 using the form **FR*x*/DBA*x*/DRV*x***, where: ▪ **FR*x*** = The number of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **DRV*x*** = The number 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. **Note:** If a drive is an exporter, it is only listed in the exporter section, see exporters on page 108.
globalSpares	The component identifiers for the drives used as Global Spare drives for the partition using the form **FR*x*/DBA*x*/DRV*x***. See **drives** above for a description of the parameters in the component identifier.
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**.

This parameter...	Specifies...
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*x*/DBA*x*/F-QIP*x***. See exporters on page 108 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, or Spectra TKLM encryption. Values: **QIPBasedBlueScale, DriveBasedBlueScale, SpectraTKLM**. - **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 a virtualized cartridge inventory to speed moves to and from slots and drives. Values: **default, virtual**. **Note:** This parameter only applies to TFinity libraries.

Syntax Error Response None.

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 and drives. The command also sets all of the other configuration parameters required for the partition. The partition is configured to allow access by all users. If you want to limit the users allowed to access the partition, modify the partition after it is created, 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's *User Guide* for detailed requirements and guidelines for configuring both data and cleaning partitions.

- Only one partition can be configured with each command.
- The library requires, at a minimum, one storage partition to be configured before you can use the library. Each storage partition must have a minimum of one storage location assigned to the storage pool and at least one drive assigned to it.
- When configuring a storage partition, all drives and controllers (RIM or F-QIP) specified in the command must already be installed in the library and must be functioning properly. If you specify a device that is not present or that is impaired, the command will fail.
- For libraries using a direct-attached drive to provide the robotic control path for a storage partition, the exporting drive counts as one drive.
- You can only create a single storage partition if the library does not have a Shared Library Services (SLS) activation key stored in the library. If you attempt to create additional storage partitions without an SLS activation, the command fails and returns an error message indicating that the command received was not a valid command for the library.

Notes:

- Cleaning partitions do not require an SLS activation key.
- If any partitions use a Fibre Channel drive to provide the robotic control path, you can configure a maximum of six drive exporters. The drive exporters can be used for six different partitions or multiple exporters can be configured in the same partition.
- If all partitions use either a RIM or QIP for the robotic control path, you can configure one partition per drive in the library up to a maximum of 16 partitions.
- See "Enabling BlueScale Software Support, Options, and Upgrades" in your library's *User Guide* for information about purchasing and entering activation keys.

- A cleaning partition can be shared by multiple storage partitions as long as the cleaning cartridges are compatible with the drives in the storage partitions.
- This command does not enable encryption. If you need 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

Notes:

- To simplify locating the corresponding information in your library's *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|Super AIT|
      Super DLT]&
   QIPExporter=[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]&
   drives=[First Drive ID]:[portaddress],...,
      [Last Drive ID]:[portaddress]&
   cleaningPartition=[partition name]&
   enablePrescan&
   enableFullscan&
   enableQuickScan=[inlineDrives|globalSpareDrives]&
   scanAfter=[time:valuewriteread],...,[time:valuewriteread]&
   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]]
```

where the value for:

This parameter...	Specifies...
partition	The name you want to use for the partition. Names can be up to 32 alphanumeric characters and can include the @ - _ /. and the space character. **Important:** The partition name is case-sensitive.
type	The type of partition to create. Values: **LTO**, **LTO Cleaning**, **TS11x0**, **TS11x0 Cleaning**, **Super DLT**, **Super AIT** **Important:** If you specify drive type that is not present, the command will fail. **Notes:** ▪ The **TS11x0** and **TS11x0 Cleaning** parameter values are only valid in a T380, T950, or TFinity library that has TS11*x*0 technology drives installed. ▪ The **Super DLT** and **Super AIT** parameters are only supported in libraries that contained these drives when purchased.
QIPExporter	The name and addressing for the controller (RIM or F-QIP) that provides the robotic control path for the partition. This controller “exports” 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. If both the **QIPExporter** and the **driveExporter** parameters are used in the same command, the command will fail. **Important:** TS11x0 technology partitions must use a QIPExporter and not a driveExporter. **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’s *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. Values: ▪ **QIP ID** = The component identifier for the exporting RIM or F-QIP using the form **FR*x*/DBA*x*/F-QIP*x***, where: ▪ **FR*x*** = The number of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **F-QIP*x*** = The number of the controller bay where the QIP is installed. For all libraries except the T120, the value of ***x*** is always **1**. For the T120 library, the value of ***x*** 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.

This parameter...	Specifies...
QIPExporter *(continued)*	▪ ***addressing mode*** = Optionally configures the addressing for each port. Only one mode can be specified in the command. If no mode is specified, the partition uses soft addressing. Values: **loop**, **fabric**, **auto**, where: ▪ **loop** = Specifies the arbitrated loop addressing mode. The loop ID is set by the value of the ***hardAddress*** parameter, if specified. If no hard address is specified, soft addressing is used. ▪ **fabric** = Specifies the fabric addressing mode. ▪ **auto** = Specifies that the addressing mode is auto-negotiated by the controller. ***hardAddress*** = The optional 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 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). 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’s *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 will fail. ⚠ **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 and later. ⚠ **Important:** A TS11x0 technology drive cannot be used as a driveExporter. TS11x0 technology 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 can support this. These multiple paths cannot be used at the same time.as long as your host software can support multiple exporting devices. Values: ▪ *Drive ID* = The component identifier for the exporting drive using the form **FR*x*/DBA*x*/DRV*x***, where: ▪ **FR*x*** = The number of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **DRV*x*** = The number 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. **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 24). ▪ 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’s *User Guide* for detailed information about drive component identifiers and addressing.

This parameter...	Specifies...
globalSpares (optional)	The component identifiers for the drives that will be used as Global Spare drives for the partition using the form **FR*x*/DBA*x*/DRV*x***. See ***DriveID*** on page 116 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 will fail. **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. ▪ 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 24). ▪ See the "Configuring and Managing Partitions" chapter in your library's *User Guide* for detailed information about configuring Global Spare drives.
numStorageSlots	The number of slots to be used for storing the cartridges that are accessible to the host for this partition. Values: **1** through ***n***, where ***n*** is the maximum number of licensed slots in the library. **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*x*0 technology 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 will fail. **Notes:** ▪ The number of slots per magazine depends on the technology. Each LTO magazine has slots for 10 LTO cartridges; a TS11*x*0 technology magazine has 9 slots. ▪ The size of the partition's storage pool is the total number of chambers you want to assign to the partition MINUS the number of chambers you want to reserve for the partition's entry/exit pool. 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, you must also subtract the number of chambers you want to use for the cleaning partition from the total number of chambers available to be assigned to the storage pool. If you have not licensed 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 be used for the partition's entry/exit pool. Values: **0** through ***n-1***, where ***n*** is the maximum number of licensed slots in the library. ⚠ **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*x*0 technology 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 will fail. **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 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...
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 will fail. Values: ▪ *Drive ID* = The component identifier for the drive using the form **FR*x*/DBA*x*/DRV*x***. See ***DriveID*** on page 116 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:** ▪ The **drives** parameter is omitted when configuring a cleaning partition. ▪ When a drive is assigned as the **driveExporter**, it 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 24). ▪ 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’s *User Guide* for detailed information about drive component identifiers and addressing.

This parameter...	Specifies...
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:** ▪ The **cleaningPartition** parameter is omitted 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.
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 "Enabling MLM and Configuring Global Settings" in your library's *User Guide*). **Notes:** ▪ The **enablePrescan** parameter is omitted when configuring a cleaning partition. ▪ The **enablePrescan** parameter is optional for storage partitions. ▪ See "Using PreScan" in your library's *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 "Enabling MLM and Configuring Global Settings" in your library's *User Guide*). ⚠ **Important:** The command cannot contain both the **enableFullscan** and the **enableQuickscan** parameters or the command will fail. **Notes:** ▪ The **enableFullscan** parameter is omitted 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's *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 all of the data on each cartridge. ⚠ **Important:** This parameter is only valid when Media Lifecycle Management (MLM) is enabled for the library (see "Enabling MLM and Configuring Global Settings" in your library's *User Guide*). ⚠ **Important:** The command cannot contain both the **enableFullscan** and the **enableQuickscan** parameters or the command will fail. Values: ▪ **inlineDrives** = Use one of the drives in the partition to verify the data on a single track of each cartridge. ▪ **globalSpareDrives** = Use a Global Spare drive assigned to the partition to verify the data on a single track of each cartridge. **Notes:** ▪ The **enableQuickscan** parameter is omitted 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. ▪ 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's *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 "Enabling MLM and Configuring Global Settings" in your library's *User Guide*). 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:** ▪ The **scanAfter** parameter is omitted when configuring a cleaning partition. ▪ The **scanAfter** parameter is omitted 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's *User Guide* for information about this parameter.

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*x*/DBA*x*/F-QIP*x***. See ***QIP ID*** on page 114 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*** = Optionally configures the addressing for each port. Values: ▪ **loop** = Specifies the arbitrated loop addressing mode. The loop ID is set by the value of the ***hardAddress*** parameter, if specified. If no hard address is specified, soft addressing is used. ▪ **fabric** = Specifies the fabric addressing mode. ▪ **auto** = Specifies that the addressing mode is auto-negotiated by the controller. When the negotiated mode is loop, the loop ID is set by the value of the ***hardAddress*** parameter, if specified. If no hard address is specified, soft addressing is used. ▪ ***hardAddress*** = The optional fixed address assigned to the port when the addressing mode is either **loop** or **auto**. Values: **0** through **125** **Notes:** ▪ The **QIPList** parameter is omitted when configuring a cleaning partition. ▪ The **QIPList** is omitted 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's *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 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:** ▪ See "Configure Mail Users" in you library's *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.

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

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

Progress Use the `partition.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>partition.xml</line>
      <line>Query string:</line>
      <line>action=new</line>
      <line>parition=[paritition name]</line>
      <line>
         type=[LTO|LTO Cleaning|TS11x0|TS11x0 Cleaning|
         Super AIT|Super DLT]
      </line>
      <line>cleaningPartition=[partition name](optional)</line>
      <line>
         QIPExporter=[[QIPID][;A[:[loop|fabric|auto]
         [:[hardAddress]]];B [:[loop|fabric|auto]
         [:[hardAddress]]]]
         </line>
      <line>
         QIPList=[QIP ID][;A[:[loop|fabric|auto][:[hardAddress]]];
         B[:[loop|fabric|auto][:[hardAddress]]]][...]
      </line>
      <line>driveExporter=[Drive ID]:[address]</line>
      <line>
         drives=[Drive ID],[Drive ID]:[A|B|AB|Hard Addressing
         Fibre ID],[...]]
      </line>
      <line>
         globalSpares=<[Drive ID],[Drive ID],[...]>(optional)
      </line>
      <line>numStorageSlots=[n]</line>
      <line>numEESlots=[m]</line>
      <line>eeType=[standard|queued|shared]</line>
      <line>
         saveLibraryConfiguration=[USB|[emailRecipient]](optional)
      </line>
      <line>enablePrescan (optional)</line>
      <line>enableFullscan (optional)</line>
      <line>
         enableQuickscan=[inlineDrives|globalSpareDrives]
         (optional)
      </line>
      <line>scanAfter=[time:N,write,read] (optional)</line>
      <line>progress (while running)</line>
   </usage>
</syntaxError>
```

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:fabric,
      FR1/DBA1/fLTO-DRV2:fabric&cleaningPartition=LTO Clean&
   enablePrescan&enableQuickScan=globalSpareDrives&
   scanAfter=write, read&
   saveLibraryConfiguration=JaneSuperuser@YourCompany.com
```

creates a single LTO storage partition named **Partition 1**. 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 the fabric addressing mode.
- 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.

CHAPTER 16

partitionList

partitionList.xml

The **partitionList.xml** command is used 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 101 for information about using the XML command interface to configure partitions in the library.

Syntax Error Response None.

Example Command and Response The following command:

```
partitionList.xml
```

returns the following list of partitions:

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

CHAPTER 17

physInventory

physInventory.xml

The **physInventory.xml** command retrieves 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 126). Partition names are case sensitive. ▪ The partition name is set when the partition is created. See partition.xml on page 101 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. **EXAMPLE:** In the example command response on page 130, the name of the partition is **Partition 1**.
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**. **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 80). **EXAMPLE:** The example command response on page 130 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. **EXAMPLE:** In the example command response on page 130, the barcode label information for the first magazine listed is **LU83567**. The barcode label information for the cartridge in slot 1 of the second magazine listed is **000380L5**.
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**. **EXAMPLE:** In the example command response on page 130, the first magazine listed is located in frame **1**.
tapeBayNumber	The number of the shelving bay within the specified frame where the magazine is located. **EXAMPLE:** In the example command response on page 130 the first magazine listed is located in shelving bay **1**.

This parameter...	Indicates...
drawerNumber	The component identifier for the chamber (drawer) where the magazine is located. **Note:** This parameter is not supported by the T120 library. **EXAMPLE:** In the example command response on page 130 the first magazine listed is located in drawer (chamber) **1**.
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:** In the example command response on page 130 the first cartridge in the second magazine listed is located in magazine slot **1**.

Syntax Error Response

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

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>
            <slot>
...
            </slot>
         </magazine>

         <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. The data can then be used as parameter values for other XML commands (for example, in the **mediaExchange.xml** command described beginning on page 75).

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 **<IE>** and **</IE>**.

   EXAMPLE See the command response data page 130 for an example of the output from **physInventory.xml** command.

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 **storag**e or **IE** 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.

   EXAMPLE The command response data for the **physInventory.xml** command shown on page 130, 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 18

powerOff

powerOff.xml

The **powerOff.xml** command power-cycles the library. See “Controlling the Library Power” in your library’s *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 backups running to the library.
2. Pause PostScan if it is running (see “Pause the PostScan Process” in your library’s *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...
numSecondsToRemainOff	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 will power 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 will remain 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>
```

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>powerOff.xml</line>
      <line>Query string:</line>
      <line>
         numSecondsToRemainOff=[number of seconds to wait
         before restoring power (0=forever)]
      </line>
   </usage>
</syntaxError>
```

Example Command and Response The following command:

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

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

CHAPTER 19

robotService

robotService.xml

The **robotService.xml** command causes a TFinity library to send a robot to service or return a robot from service.

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 will fail.

> **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]`

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.

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

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

Progress Use the `robotService.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>robotService.xml</line>
      <line>Query string:</line>
      <line>action=returnFromService</line>
      <line>robot=[robot number]</line>
      <line>progress (while running)</line>
   </usage>
</syntaxError>
```

sendTo Service

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

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 will reinitialize. If the robot has been 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]`

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.

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

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

Progress Use the `robotService.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>robotService.xml</line>
      <line>Query string:</line>
      <line>action=sendToService</line>
      <line>robot=<robot number></line>
      <line>progress (while running)</line>
   </usage>
</syntaxError>
```

CHAPTER 20

robotUtilization

robotUtilization.xml

The **robotUtilization.xml** command lets you monitor the percentage of each hour that the library's 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.

Syntax Error Response None

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 21

systemMessages

systemMessages.xml

The **systemMessages.xml** command retrieves all system messages currently stored on the library. See “Check and Respond to Messages” in your library’s *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. **EXAMPLE:** In the command response below, the first message has a **severity** value of **Info**; the second message has a **severity** value of **Warning**.
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.

Syntax Error Response None

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>2011</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>2011</year>
      </date>
      <time>
         <hour>16</hour>
         <minute>19</minute>
         <second>08</second>
      </time>
      <notification>AutoSupport is not configured to automatically
         email data about critical events and errors, though this
         functionality is supported.</notification>
      <remedy>Using the library interface, select AutoSupport ->
         Manage Events and enable critical AutoSupport events.
      </remedy>
   </message>
...
</systemMessages>
```

CHAPTER 22

taskList

taskList.xml

The **taskList.XML** command returns a list of the operations currently being processed by the library.

[no parameters]

Description Returns a list of the asynchronous actions and background operations currently being processed by the library. See Progress for Asynchronous Commands on page 15 for additional information about how the library handles asynchronous actions.

Syntax `taskList.xml`

Command Response

- If the library is not currently performing any asynchronous or background operations, the command returns a page with empty parameter tags.
- If the library is performing any asynchronous 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>
         </thread>
      </task>
...
      <task>
...
      </task>
   </currentBackgroundTasks>
</taskList>
```

where the value for:

This parameter...	Indicates...
current Asynchronous Action	The section of the command response that contains information about any asynchronous operation the library is processing. **Notes:** ▪ The library only processes one asynchronous operation at a time. ▪ If the library is not currently processing any asynchronous operations, this section will appear 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 will appear 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 being processed.
name	The name of the asynchronous action or background task currently being processed.
status	The current status of the asynchronous action. Values: ▪ **QUEUE** = The asynchronous action is queued for processing. ▪ **SUBMITTED** = The asynchronous action has been submitted for processing. ▪ **ACTIVE** = The asynchronous action is being processed. ▪ **WAITINGFORUSER** = The asynchronous action is waiting for the operator to respond to the feedback string. ▪ **DETACHED** = The asynchronous 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 feedback is required from the operator before processing of the asynchronous task can continue. **Note:** The **feedbackString** parameter is only included if the status is **WAITINGFORUSER**.
thread	The description of each background client involved in processing the background operation indicated by the **name** parameter in the task section of the response data.

Syntax Error Response None

Example Command and Response The following command:

```
taskList.xml
```

```
<taskList>
   <currentAsynchronousAction>
      <name>MOVE_ELEMENTS</name>
      <status>ACTIVE</status>
   </currentAsynchronousAction>
</taskList>
```

CHAPTER 23

traces

traces.xml

The **traces.XML** command retrieves the CAN logs, QIP logs, and other traces stored on the library's LCM. Spectra Logic uses these logs and traces to help diagnose problems with the library. You only need to capture traces when instructed to do so by Spectra Logic Technical Support. See "Capturing Traces" in your library's *User Guide* for additional information about the traces and logs.

getCanLog Names

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** command 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. The zip file is created at the time it is requested. ▪ **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.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>traces.xml</line>
      <line>Query string:</line>
      <line>
         action=<getCanLogNames|getCanLog|
         getQIPLogNames|getQIPLog>
      </line>
   </usage>
</syntaxError>
```

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 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.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>traces.xml</line>
      <line>Query string:</line>
      <line>action=getCanLog</line>
      <line>name=<canLogName></line>
   </usage>
</syntaxError>
```

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.

getQIPLog Names

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** command 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.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>traces.xml</line>
      <line>Query string:</line>
      <line>
         action=<getCanLogNames|getCanLog|
         getQIPLogNames|getQIPLog>
      </line>
   </usage>
</syntaxError>
```

Example Command and Response The following command 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 to determine the names of the QIP logs currently stored on the hard drive in the LCM.

⚠ **Important** The **getQIPLog** command 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.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>traces.xml</line>
      <line>Query string:</line>
      <line>action=getQIPLog</line>
      <line>name=<qipLogName></line>
   </usage>
</syntaxError>
```

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.

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’s *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*]** ▪ **SNMP** ▪ **SpectraTKLM** ▪ **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*x*/DBA*x*/F-QIP*x***, where: ▪ **FR*x*** = The number of the frame. Only used in the component identifier when the controller is in a library that supports multiple frames. ▪ **DBA*x*** = The number of the drive bay assembly (DBA) containing the controller. Not used with the T120 library. ▪ **F-QIP*x*** = The interface type of the controller (Fibre Channel) and the number of the controller bay where it is installed. For all libraries except the T120, the value of ***x*** is always **1**. For the T120 library, the value of ***x*** 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.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>traces.xml</line>
      <line>Query string:</line>
      <line>traceType=[Action|AutoDriveCLean|AutoSupport]</line>
      <line>traceType=[BackgroundClient|CAN|Connection]</line>
      <line>traceType=[Encryption|Error|EtherLib|Event]</line>
      <line>traceType=[GPIO|HHM|Inventory|Kernel|Lock]</line>
      <line>traceType=[Message|MLM|PackageUpdate|SNMP]</line>
      <line>traceType=[SpectraTKLM|WebServer]</line>
      <line>traceType=[Initialization|LogicalLibrary|Motion]</line>
      <line>traceType=[HydraExit|Geometry|Pools]</line>
      <line>traceType=[MotionInventory|MotionOptions]</line>
      <line>traceType=[MotionRestart1|MotionRestart2]</line>
      <line>traceType=[QIP:name of QIP to act on]</line>
      <line>traceType=[QIPDump:name of QIP to act on]</line>
   </usage>
</syntaxError>
```

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 24

utils

utils.xml

The **utils.xml** command provides utilities used for troubleshooting and maintaining the library.

lockTension Rods

Description Changes whether the tension rods are locked or can disengage when it is necessary for one TeraPorter in a TFinity library to push the other TeraPorter into it's service bay.

> **Note:** Do not change the behavior of the tension rods unless specifically instructed to do so by Spectra Logic Technical Support.

Syntax `utils.xml?action=lockTensionRods&state=[on|off]`

where the value for:

This parameter...	Indicates...
state	Whether the tension rods are locked. Values: ▪ **on** = The tension rods are always locked and will not disengage. This is the libraries default behavior. ▪ **off** = The tension rods are unlocked and can disengage as necessary.

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

```
<utils>
   <lockTensionRods>
      <status>OK</status>
   </lockTensionRods>
</utils>
```

Progress Use the `utils.xml?progress` command to determine when the reset operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>utils.xml</line>
      <line>Query string:</line>
      <line>action=lockTensionRods</line>
      <line>state=<on|off></line>
      <line>progress(while running)</line>
   </usage>
</syntaxError>
```

removeAll Library Partitions

Description Removes all partitions configured in the library and then cycles library power.

Syntax `utils.xml?action=removeAllLibraryPartitions`

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

```
<utils>
   <removeAllLibraryPartitions>
      <status>OK</status>
   </removeAllLibraryPartitions>
</utils>
```

When the command completes, 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 the `utils.xml?progress` command to determine when the operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15) or there is no response from the library.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>utils.xml</line>
      <line>Query string:</line>
      <line>action=[lockTensionRods|removeAllLibraryPartitions|
         resetLCM|resetRobot|resetInventory|selectiveSnowplow|
         verifyMagazineBarcodes]</line>
      <line>progress(while running)</line>
   </usage>
</syntaxError>
```

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 resets the RCM in libraries that use an RCM.
- This command 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`

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

```
<utils>
   <resetInventory>
      <status>OK</status>
      <message>
         resetInventory started.
         Set progress in your query for status
      </message>
   </resetInventory>
</utils>
```

Progress Use the `utils.xml?progress` command to determine when the reset operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>utils.xml</line>
      <line>POST data:</line>
      <line>action=[lockTensionRods|removeAllLibraryPartitions|
         resetLCM|resetRobot|resetInventory|selectiveSnowplow|
         verifyMagazineBarcodes]</line>
      <line>progress(while running)</line>
   </usage>
</syntaxError>
```

resetLCM

Description Resets the LCM, which resets the front panel and the web server.

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. Backup operations are not affected.

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>utils.xml</line>
      <line>Query string:</line>
      <line>action=[lockTensionRods|removeAllLibraryPartitions|
         resetLCM|resetRobot|resetInventory|selectiveSnowplow|
         verifyMagazineBarcodes]</line>
      <line>progress(while running)</line>
   </usage>
</syntaxError>
```

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 command is only supported on T950 libraries.

Syntax `utils.xml?action=resetRobot`

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

```
<utils>
   <resetRobot>
      <status>OK</status>
   </resetRobot>
</utils>
```

Progress Use the `utils.xml?progress` command to determine when the reset operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>utils.xml</line>
      <line>Query string:</line>
      <line>action=[lockTensionRods|removeAllLibraryPartitions|
         resetLCM|resetRobot|resetInventory|selectiveSnowplow|
         verifyMagazineBarcodes]</line>
      <line>progress(while running)</line>
   </usage>
</syntaxError>
```

selective Snowplow

Description Changes 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 command is only supported on TFinity libraries.

Syntax `utils.xml?action=selectiveSnowplow&state=[on|off]`

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.

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

```
<utils>
   <selectiveSnowplow>
      <status>OK</status>
   </selectiveSnowplow>
</utils>
```

Progress Use the `utils.xml?progress` command to determine when the reset operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>utils.xml</line>
      <line>Query string:</line>
      <line>action=selectiveSnowplow</line>
      <line>state=<on|off></line>
      <line>progress(while running)</line>
   </usage>
</syntaxError>
```

verify Magazine Barcodes

Description Runs the advanced utility to verify the barcodes on all TeraPack magazines.

> **Note:** This command is supported on T200, T380, T680, T950, and TFinity libraries.

Syntax `utils.xml?action=verifyMagazineBarcodes`

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

```
<utils>
   <verifyMagazineBarcodes>
      <status>OK</status>
      <message>verifyMagazineBarcodes started. Set progress in
         your query for status.</message>
   </verifyMagazineBarcodes>
</utils>
```

Progress Use the `utils.xml?progress` command to determine when the reset operation is complete. The operation is complete when the command response is **No Pending Actions** (see Progress for Asynchronous Commands on page 15).

Syntax Error Response

```
<syntaxError>
   <message>[error message text]</message>
   <usage>
      <line>utils.xml</line>
      <line>Query string:</line>
      <line>action=[lockTensionRods|removeAllLibraryPartitions|
         resetLCM|resetRobot|resetInventory|selectiveSnowplow|
         verifyMagazineBarcodes]</line>
      <line>progress(while running)</line>
   </usage>
</syntaxError>
```

INDEX