STAR XML Developer Manual

April 2015

CUADRA ASSOCIATES, INC.

3415 S. Sepulveda Blvd., Suite 210, Los Angeles, CA 90034 ● (310) 591-2490 ● Fax (310) 591-2488
E-mail: support@cuadra.com ● Internet: www.cuadra.com

Table of Contents

1. Introduction

1.1 Overview

The STAR information storage and retrieval system can be accessed through a number of interfaces. The interfaces known as **STAR Client**, **STAR Web**, **classic STAR**, and **STAR Z39.50** are for use by people, called end users, who use these interfaces to interact with STAR for searching, report generation, data entry, database maintenance, and system maintenance.

In contrast, the interfaces known as **STAR ADO** and **STAR XML** are **application programming interfaces** (**API**s), which are for use by programs. These interfaces provide other programs with access to the same STAR functions that the end user interfaces provide, so that other programs can take advantage of STAR's features.

Some programs that come with STAR rely on STAR APIs. You can write your own programs that use STAR APIs and you can add STAR access features to your own existing programs using STAR APIs.

STAR XML is a STAR interface that provides access to one or more STAR systems, using XML documents for communication. STAR XML runs as a server application, named **STAR XML Server**, on a **STAR XML host**. The STAR XML host must have the appropriate **Java Runtime Environment** software (and must therefore be a platform that supports Java). It can be, but does not have to be, a STAR host as well. A single STAR XML Server can communicate with any number of client programs and with any number of STAR systems. STAR XML can be installed on any number of hosts in your network.

Extensible Markup Language (**XML**) is an industry-standard language for the exchange of structured documents and data. Adoption of XML technology is growing rapidly due to the wide availability of XML tools, integrated support for XML in the latest web browsers, and the backing of Microsoft, IBM, and other major manufacturers.

With STAR XML, requests for STAR services are made using commands that form the data in an XML document. STAR XML Server processes the command requests, adds the results to the XML document, and returns the resulting XML document. The structure of these XML documents allows them to contain tagged commands, tagged parameters, and tagged results. The command language is formally defined by an XML **Document Type Definition** (**DTD**).

Requests to STAR can be submitted, via TCP/IP, from any client program running on any computer that is networked to the STAR host (*see* Figure 1). Programs can format requests, extract results, or do other processing of the XML files using standard XML routines and techniques, e.g., using widely available XML parsers and XML constructors. XML documents can also be displayed directly by some web browsers and can be edited using XML editors.

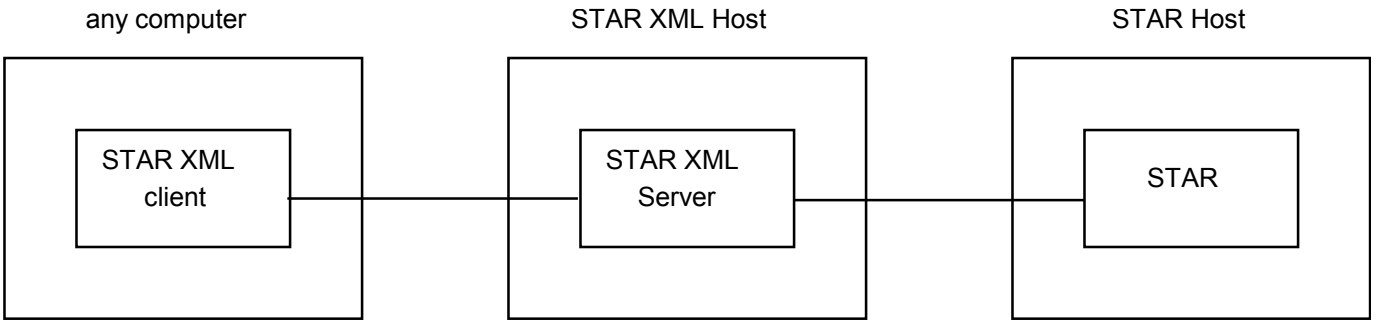

Figure 1: STAR XML Architecture

Because STAR XML Server adds STAR results to the XML request document and returns it as the XML response document, the XML response document is always a superset of the XML request document. In other words, no elements, attributes, or data in the input document are changed. The output document consists of the input document with additional elements, attributes, and data representing the responses from STAR. However, there may be superficial differences between the XML request document and the XML response document, such as tabs changing to spaces, double quote marks changing to their """ form, and changes in the order of an element's attributes.

Any program that uses STAR XML by submitting a request to a STAR XML host is called a **STAR XML client program**. As used in this manual, the lowercase "client" in this phrase distinguishes it from the specific program named **STAR XML Client** that is provided with STAR XML.

STAR XML client programs can be written in any language, can run on any platform, and can be used for any purpose. For example, a client program could be a standalone program, a program called from another program or a script, or a cgi-bin program invoked by a web server. Two sample STAR XML client programs, **STAR XML Client** and **starxml**, are provided with STAR XML.

STAR XML does not require particular STAR databases or particular database fields or database definitions. It can be used with any STAR database. However, when you are writing a STAR XML client program for a particular purpose, you may find it useful to define any of the following specifically for use by your program:

- STAR search fields
- STAR output fields
- STAR report formats
- STAR pagelayouts
- STAR assisted search specifications
- STAR saved searches
- STAR global records
- STAR printers and pseudo printers
- STAR license reservations

1.2 What Comes with STAR XML

STAR XML ships as part of STAR although STAR XML Server can be run on any suitable host, whether or not it is a STAR host. When the STAR XML host will not be the STAR host, the necessary files can be copied from the STAR host to the STAR XML host.

The following STAR XML files can be found in the STAR XML directory:

STARXMLServer.jar: An executable Java program named **STAR XML Server**. This is the main program that makes STAR XML available. It is run on one or more STAR XML hosts, to which STAR XML client programs connect.

STARXMLClient.jar: An executable Java program named **STAR XML Client**. This is a simple graphical program that allows you to open and display an XML request document from disk, submit it to a STAR XML host, and then display and/or save to disk the XML response document. This program runs on any host that has the Java Runtime Environment.

STARXMLRequest.dtd: The Document Type Definition of STAR XML request documents. This defines the elements and attributes of a STAR XML request and specifies their names, values, whether they are required or optional, etc.

STARXMLResponse.dtd: The Document Type Definition of STAR XML response documents. This defines the elements and attributes of a STAR XML response, and specifies their names, values, whether they are required or optional, etc. It is a superset of **STARXMLRequest.dtd**.

sample_request.xml: A sample STAR XML request document, showing a variety of STAR XML commands. The **STAR XML Client** program or **starxml** command can be used to submit this document to STAR XML. File **sample_request.xml** is listed in Appendix D.

sample_response.xml: A sample STAR XML response document, corresponding to STAR XML request document **sample_request.xml**. File **sample_response.xml** is listed in Appendix D.

starxml (Unix) or **starxml.exe** (Windows): A shell-level command that submits an XML request to STAR XML and outputs the resulting XML response. This program runs on the STAR host and can be used within shell scripts.

xmltest: A shell-level command for testing STAR XML connections. This program is used during STAR XML installation and can be used later to test that STAR XML connections still work.

The following STAR XML files are available to authorized STAR XML users in STARFISH, the downloading system at the Cuadra Associates web site:

xmldev.pdf: The **STAR XML Developer Manual** (this manual) as a PDF file.

xmlref.pdf: The **STAR XML Reference Manual** as a PDF file. The manual explains the meaning and use of each of the STAR XML commands.

1.3 STAR XML and STAR Licensing

Purchase of a STAR system and possession of a STAR license agreement are required for each STAR system you have. Each STAR system is licensed for a particular maximum number of simultaneous sessions.

In contrast, no fee or license is needed to run STAR XML Server or any STAR XML client program. You may install and run these on as many hosts as you like. The number of active STAR XML sessions is unlimited in theory, although there may be limits in practice based on the system resources (particularly memory) available to the Java Runtime Environment (JRE).

However, when STAR XML is used to access a STAR host, STAR XML Server requires one STAR session on that STAR system while the commands from a STAR XML client program are being executed. These sessions count toward the limit on the number of simultaneous sessions for which the STAR system is licensed and therefore limit the maximum number of STAR XML client programs that can be interacting with a given STAR host simultaneously. STAR XML sessions on a STAR system can be identified using **STAR Manager** or the **starsys users** and **starsys sessions** commands on the STAR host.

2. Using STAR XML—General Principles

2.1 What a STAR XML Client Program Does

The STAR System (on the STAR host) and STAR XML Server (on the STAR XML host) must be running before STAR XML client programs can use STAR XML. For instructions on starting STAR XML Server, *see* Appendix A.

A typical STAR XML client program performs these steps:

1. Opens a two-way connection to STAR XML Server on the STAR XML host via a TCP/IP socket.
2. Constructs, on disk or in memory, a well-formed and valid XML request document containing one or more requests to STAR.
3. Writes the XML request document to STAR XML Server, followed by a Control-Z character (hex 1A). STAR XML requests are always sent one document at a time. Control-Z is used as an end-of-file indicator. (An actual end-of-file cannot be used to indicate the end of a request document because the socket must remain open to receive the response.)
4. Reads the XML response document from STAR XML Server. Results are always returned one document at a time. The response will also be followed by a Control-Z character (hex 1A). The XML response document will be a superset of the XML request document (except that spacing, quoting, and the order of an element's attributes may change).
5. Interprets the STAR results contained in the XML response document.

6. Repeats Steps 2, 3, 4, and 5 as many times as necessary. If no more repetitions are to be made, the last XML request document can end with the **Quit** command, which causes STAR XML to close the connection.
7. Closes the connection to the STAR XML host.

If a STAR XML client program will make many requests to STAR (e.g., multiple searches, reports, updates, etc.), they can be combined in a single XML request document or they can be divided into multiple documents, each with one or a few requests. The advantage of using multiple documents is that the program can decide what requests to make based on responses to previous requests. The entire connection, during which one or more XML request documents are processed, is called a **session**.

Example: Using an XML request document, a STAR XML client program might retrieve a set of STAR records from a given database, using the search **PENDING=YES**, check that one or more records were retrieved, and generate a list of their record numbers. Using another XML request document (i.e., a second request to the same STAR XML Server), it might then perform updates to each of the retrieved records.

Rather than do its own communication with STAR XML, a shell script or program that runs on the STAR host can use the **starxml** command (*see* Appendix C) to handle the communication with STAR XML.

2.2 STAR XML Files

STAR XML request documents and STAR XML response documents are XML documents and must conform to the rules of XML document syntax and structure. In particular, all container element tags (e.g., <**Search**>) must have matching end tags (e.g., <**/Search**>), all empty (non-container) element tags must end with a slash (e.g., <**Quit/**>), and all attribute values must be quoted (e.g., **Host="superstar"**). An XML document that conforms to these rules is said to be **well-formed**. The rules of XML are much stricter than the rules of HTML, where omitted tags, mismatched tags, and omitted quote marks are permitted.

The following is a well-formed XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<Planets>
	<Planet name="Mercury">
		<Composition>rocks</Composition>
		<Temperature>hot</Temperature>
		<Stats diameter="4880km" orbit="58Mkm"/>
	</Planet>
	<Planet name="Earth">
		<Composition>rocks</Composition>
		<Temperature>warm</Temperature>
		<Stats diameter="12756km" orbit="150Mkm"/>
	</Planet>
	<Planet name="Neptune">
		<Composition>gas</Composition>
		<Temperature>cold</Temperature>
		<Stats diameter="49500km" orbit="4497Mkm"/>
	</Planet>
</Planets>
```

STAR XML request documents must also follow the more specific rules of STAR XML requests, as defined in file **STARXMLRequest.dtd**. These rules define the elements and attributes that are permitted. An XML document that conforms to a given Document Type Definition is said to be **valid**.

The XML document above, while well-formed, is not a valid STAR XML request document. The following STAR XML document is both well-formed and valid:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
      <Login>
            <Connection Id="con1" Host="jupiter" UserId="sarah"/>
      </Login>
      <Database Id="db1" ConnectionId="con1" DatabaseName="DOCS"/>
      <Report Id="rpt1" DatabaseId="db1" PageLayout="STARXML"
      ReportName="browse">
            <SearchText>R=123</SearchText>
      </Report>
      <GetRecords ReportId="rpt1" FirstRecord="1"/>
      <Quit/>
</Root>
```

STAR XML response documents are always well-formed and are always valid according to the Document Type Definition in file **STARXMLResponse.dtd**.

The spacing and indenting of elements and attributes in an XML file do not affect its meaning. Examples in this manual are shown with indenting that mirrors the nesting of elements, which is always a good habit for making XML documents more readable and the process of writing them less error-prone. However, STAR XML documents will usually be generated and processed by programs, not people, so readability is useful for troubleshooting but is not a concern for routine use of STAR XML.

If the STAR host is using Latin-1 (ISO 8859-1) as set by the **CHARSET** option in file *~star/config/starsys.opt* then STAR XML request documents can specify either **encoding="UTF-8"** or **encoding="ISO-8859-1"**. If the STAR host is using Latin-2 (ISO-8859-2) then STAR XML request documents can specify either **encoding="UTF-8"** or **encoding="ISO-8859-2"**. STAR XML will use the same encoding in the XML response document.

UTF-8 encoding allows all Unicode characters to be represented, some in one byte and some with multi-byte sequences. If you specify **encoding="UTF-8"** then STAR converts data received from STAR from Latin-1 or Latin-2 to UTF-8 and converts data sent to STAR back to Latin-1 or Latin-2. In UTF-8 encoding you may not use non-ASCII Latin-1 or Latin-2 characters in one-byte form.

If you specify **encoding="ISO-8859-1"** or **encoding="ISO-8859-2"** then characters that are not in Latin-1 or Latin-2 cannot be specified in one-byte or multi-byte form and must instead rely on STAR ^D codes or HTML entity representation.

Example: Choices for character representation

		Encoding		
Character	**Representation**	**ISO-8859-1**	**ISO-8859-2**	**UTF-8**
é (acute accent e)	direct	hex E9	hex E9	hex CE hex A9
	entity	é or **é**		
	^D code	**^D'e**		
ć (acute accent c)	direct	(can't represent)	hex E6	hex C4 hex 87
	entity	**ć**		
	^D code	**^D'c**		
£ (pound)	direct	hex A3	(can't represent)	hex C2 hex A3
	entity	&pound or **£**		
	^D code	**^D$L**		
Ω (uppercase Omega)	direct	(can't represent)	(can't represent)	hex CE hex A9
	entity	&Omega or **Ω**		
	^D code	**^D*W**		

Following the rules of XML syntax, certain characters within the body data of a STAR XML request document must be encoded as "entity references" (or in numeric **&#nnn;** form) to prevent ambiguities. In particular, the less-than sign (<), greater-than sign (>) and ampersand (&) must be encoded as follows:

< for <
> for >
& for &

Other characters can be, but do not need to be, encoded, e.g.,

' for '

" for "

Other HTML entity references can be used as well. Signed characters in Latin-1 or other 8-bit character sets should be represented with entity references, in numeric form, or encoded as UTF-8. In particular, a signed Latin-1 character must not be represented using its one-byte value, since that is not a valid encoding of the character in UTF-8.

Examples:

For	**non-breaking space**	use either	** **	or	** **	or	UTF-8: **hex C2 A0**
For	**registered mark**	use either	**®**	or	**®**	or	UTF-8: **hex C2 AE**
For	**á**	use either	**á**	or	**á**	or	UTF-8: **hex C3 A1**

Errors:

For	**non-breaking space**	do not use	Latin-1: **hex A0**
For	**registered mark**	do not use	Latin-1: **hex AE**
For	**á**	do not use	Latin-1: **hex E1**

2.3 STAR XML Commands

The actions in an XML request document are specified as a sequence of commands. The commands can be categorized as follows:

Session control	commands to start and end a session, get information about the servers involved, and identify needed resources.
Search and report	commands to perform STAR's main search and report functions.
Data entry	commands to create, update, delete, lock, and unlock database records.
Database maintenance	commands to operate on databases, views, and database definitions.
Space maintenance	commands to operate on Spaces.

Each command is a sub-element of the root element and may have both required and optional attributes. Element and attribute names are case sensitive.

When a STAR XML client program opens a connection to a STAR XML host, the first XML request document must begin with a **Login** command. The **GetSTARXMLInfo** command is an exception; it can appear anywhere, with or without a previous **Login**. In the remainder of the XML request document, and in any further request documents sent, all STAR XML commands other than **Quit** can be used. The last XML request document can end with a **Quit** command to close the connection. If it does not, STAR XML Server will wait for more requests or until the connection is closed by the STAR XML client program.

For purposes of STAR XML, the words **command** and **element** mean essentially the same thing, both referring to XML elements used to specify STAR XML commands. Similarly, the words **parameter**, **flag**, and **attribute** mean essentially the same thing, all referring to the use of XML attributes to specify options and control features of STAR XML commands.

The STAR XML commands are listed both alphabetically and within categories in the STAR XML Reference Manual.

Over time, STAR XML commands and their syntax and features may change. Documentation of these changes will make clear which versions of STAR or STAR XML are first to support each change. That way, STAR XML client programs can use the **GetSTARXMLInfo** and **GetConnectionInfo** commands to check the server versions so they can avoid using new commands or features under an older version of STAR or STAR XML.

Example: Retrieving the STAR XML Server version number:

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
       <GetSTARXMLInfo/>
       <Quit/>
</Root>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root ecount="0">
       <GetSTARXMLInfo Version="1.0"/>
       <Quit/>
</Root>
```

Example: Retrieving the STAR version number:

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
       <Login>
              <Connection Id="mycon" Host="galaxy" UserId="frontdesk"
              Password="xyz"/>
       </Login>
       <GetConnectionInfo ConnectionId="mycon"/>
       <Quit/>
</Root>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root ecount="0">
       <Login>
              <Connection Id="mycon" Host="galaxy" UserId="frontdesk"
              Password="xyz"/>
       </Login>
       <GetConnectionInfo ConnectionId="mycon" Version="3.8.32"
       Charset="8859-1"/>
       <Quit/>
</Root>
```

In the example above and other examples in this manual, the elements and attributes added by STAR XML in an XML response document are shown underlined.

2.4 Resources and IDs

As the commands in an XML request document are executed by STAR XML Server, specific types of resources are allocated, each identified by a unique string called an **ID**. Each ID is specified in the XML request document and is used to refer to the resource in subsequent commands. In other words, IDs are selected by the program making the request, not by STAR XML Server. For example, a **DatabaseId** will be specified for each STAR database to be used in a given XML request document, and the **DatabaseId** will be used to identify the database in commands for searching, reporting, updating records, etc. in that database.

The resources that are given ID strings are as follows:

Connection	identifies a connection to a STAR session on a given STAR host.
Database	identifies a particular STAR database on a given STAR host.
DatabaseGroup	identifies a collection of STAR databases on one or more STAR hosts. Used to perform a single search and/or report operation on a set of databases.
Report	identifies a STAR report.
Lock	identifies the lock that must be acquired on a new or existing database record before the record can be created or updated.

The resulting ID strings have attribute names **ConnectionId**, **DatabaseId**, **DatabaseGroupId**, **ReportId**, and **LockId**.

Any number of IDs can be used within a STAR XML session, but each ID represents memory and/or disk space in use on the STAR host(s). If the same string is used for IDs of different types (e.g., a **ConnectionId** and a **DatabaseId**), they are independent of each other, i.e., each type has its own namespace. A given ID of a given type can be reused within a session, which will free the previous memory or disk space used by the resource and assign the ID to the new resource.

Resources are retained during a session even when multiple request documents are submitted. For example, if a request contains a **Login** command that establishes a Connection ID and does not contain a **Quit** command, a subsequent request can continue to reference that Connection ID.

Example: In the XML request document below, the sole connection is given an ID of **My1stConnection** and the two databases are given IDs of **MainDatabase** and **OtherDatabase**:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
      <Login>
            <Connection Id="My1stConnection" Host="jupiter" Port="11002"
            UserId="sarah" Password="wink"/>
      </Login>
      <Database Id="MainDatabase" ConnectionId="My1stConnection"
      DatabaseName="NEWDOCS"/>
      <Database Id="OtherDatabase" ConnectionId="My1stConnection"
      DatabaseName="EXTRAS"/>
      <!-- Commands to use the databases would appear here. -->
      <Quit/>
</Root>
```

The same example could just as well have used the host name and database names as the IDs, e.g.,

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
	<Login>
		<Connection Id="jupiter" Host="jupiter" Port="11002"
		UserId="sarah" Password="wink"/>
	</Login>
	<Database Id="newdocs" ConnectionId="jupiter" DatabaseName="NEWDOCS"/>
	<Database Id="extras" ConnectionId="jupiter" DatabaseName="EXTRAS"/>
	<!-- Commands to use the databases would appear here. -->
	<Quit/>
</Root>
```

2.5 Passwords

In order to make a connection to a STAR system, a STAR XML client program will need a STAR User ID and the corresponding STAR password. They are used with the **Login** command. The STAR User ID can be one that is used with other STAR interfaces or one defined specifically for use with STAR XML.

When STAR databases are password protected, an XML request document must specify the necessary passwords for the level of access required. The four levels are:

Search

Data Entry (validated update)

Global (unvalidated update)

Database Definition

These passwords can be specified when a database is first opened or can be specified with the commands for certain individual database operations. For each database, STAR keeps track of the maximum password level specified so far and allows database operations only if a password of at least the necessary level has been specified during the session.

Any necessary **Search** password should be specified when a database is opened, because it cannot be specified with the **Search**, **Report**, and **ShowIndex** commands. These commands do not provide a **Password** attribute because they operate on both databases and database groups, so a single password would not always suffice.

Example: A search password is specified with the **Database** command for use with subsequent **Report** and **GetRecords** commands. A **DBDEF** password is specified directly with the **SaveDatabase** command.

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
	<Login>
		<Connection Id="con1" Host="jupiter" UserId="sarah"/>
	</Login>
	<Database Id="db1" ConnectionId="con1" DatabaseName="DOCS"
	Password="readok"/>
	<Report Id="rpt1" DatabaseId="db1" PageLayout="STARXML"
	ReportName="browse">
		<SearchText>R=123</SearchText>
	</Report>
	<GetRecords ReportId="rpt1" FirstRecord="1"/>
	<SaveDatabase ConnectionId="con1" DatabaseName="DOCS"
	Password="dbmanage" FileName="docs.img"/>
	<Quit/>
</Root>
```

2.6 Case Sensitivity

Most names that are uppercase in classic STAR can be specified in any case in STAR XML, while certain elements of an XML request document are case sensitive. Refer to the lists below for specific items. For items not covered, assume that names and strings are case sensitive.

Case insensitive items:

- database names
- database passwords
- GLOBAL database record names
- host names
- input field names
- report format names
- search expressions
- search field names
- search line values
- SERVER database record names
- SERVER database search names
- Space names
- STAR passwords
- STAR User IDs

Case sensitive items:

- attribute tags in XML documents
- Connection ID strings
- Database ID strings
- DatabaseGroup ID strings
- element tags (command names)
- filenames
- global variable names
- global variable values
- keywords, e.g., attribute values "True" and "False"
- Lock ID strings
- output field names
- Report ID strings

2.7 Error Handling

When an XML request document is submitted, STAR XML attempts to execute all commands specified in the document. Some or all of them may succeed, and some or all of them may fail. Failure may result, for example, if a host system is not reachable, a database is not available, or a required password is missing or specified incorrectly. Errors can also result from syntax errors in the XML request document itself.

STAR XML returns an XML response document whether or not errors occur.

For a request document that is not well-formed (has invalid elements or attributes or does not follow XML syntax conventions), STAR XML returns the following XML result document:

```
<?xml version="1.0" encoding ="UTF-8"?>
<Root ecount="1" emsg="Request not well-formed"/>
</Root>
```

For a well-formed request document, each command element that produced an error is given an attribute named **ecount** and an attribute named **emsg**. The value of the **ecount** attribute is the number of errors that occurred while executing this command. The value of the **emsg** attribute is the text of an error message describing the problem. Any command other than **GetSTARXMLInfo** and **Quit** has the potential to produce an error result. The **GetSTARXMLInfo** and **Quit** commands never fail.

Many commands return additional attributes and/or subelements. For example, the **CreateLock** command can return a **RecordNumber** attribute and the **GetRecords** and **ShowIndex** commands can return a subelement named **Response**. These added attributes and subelements can be thought of as STAR XML's "output data." (This output data is what is shown underlined in examples.) When errors occur, the **ecount** and **emsg** attributes are returned, and possibly other output data as well. When errors do not occur, output data is returned for all applicable attributes other than **ecount** and **emsg**. This is why, for example, the STAR XML Reference Manual and the STAR XML DTDs show the **RecordNumber**, **ecount**, and **emsg** attributes of the **CreateLock** command as optional, even though at least one will always be returned.

The **ecount** attribute is added to a command only when errors occur, except at the root level of the XML response document, where an **ecount** attribute is always returned. Therefore, when examining an XML response document, a STAR XML client program can detect overall success by checking that the top-level **ecount** attribute has a value of zero and/or checking for the absence of **ecount** attributes in individual commands. Similarly, it can detect failure by checking that the top-level **ecount** attribute has a value greater than zero and/or checking for the presence of **ecount** attributes in individual commands.

If a single command generates more than one error, **ecount** will be more than 1 and **emsg** will contain multiple error messages, separated by semicolons.

Certain types of errors, such as a failed **Login** command, will cause all subsequent commands to fail. Others, such as an **UpdateRecord** that failed, may have no effect on the success of subsequent commands.

If a STAR XML client program wants to execute some commands conditionally based on the success of, or the results of, previous commands, it should submit an XML request document with the first set of commands, examine the results (including the **ecount** attributes), and then decide whether to submit another XML request document with additional commands.

Example: Referencing a good database and a bad database:

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
	<Login>
		<Connection Id="con" Host="jupiter" UserId="sarah"/>
	</Login>
	<Database Id="1" ConnectionId="con" DatabaseName="FINE"/>
	<Database Id="2" ConnectionId="con" DatabaseName="OOPS"/>
	<Report Id="rpt1" DatabaseId="1" PageLayout="STARXML"
	ReportName="*DUMP">
		<SearchText>MONTH=SEPTEMBER</SearchText>
	</Report>
	<Report Id="rpt1" DatabaseId="2" PageLayout="STARXML"
	ReportName="*DUMP">
		<SearchText>MONTH=SEPTEMBER</SearchText>
	</Report>
	<Quit/>
</Root>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root ecount="2">
      <Login>
            <Connection Id="con" Host="jupiter" UserId="sarah"/>
      </Login>
      <Database Id="1" ConnectionId="con" DatabaseName="FINE"/>
      <Database Id="2" ConnectionId="con" DatabaseName="OOPS"
      ecount="1" emsg="starapi.SSException; DBNotFound;  "/>
      <Report Id="rpt1" DatabaseId="1" PageLayout="STARXML"
      ReportName="*DUMP" LineCount="5" RecordCount="1">
            <SearchText>MONTH=SEPTEMBER</SearchText>
      </Report>
      <Report Id="rpt1" DatabaseId="2" PageLayout="STARXML"
      ReportName="*DUMP" ecount="1"
      emsg="starapi.SSException; badDatabaseId: 2;  ">
            <SearchText>MONTH=SEPTEMBER</SearchText>
      </Report>
      <Quit/>
</Root>
```

2.8 Global Variables

The **SetVariable** command lets you set global variables that apply to your STAR XML session. The variables are passed to each STAR system used in your session. They are often referred to as "dollar variables" because references to them are preceded by a dollar sign.

Five ways in which global variables can be used within a STAR XML session are shown below. Refer to the relevant sections of this manual and the STAR XML Reference Manual for details about the specific commands used.

Global variables can be used:

- in the search expression within the **SearchText** element of the **Report** command, to use the value of the global variable within the search. Note that if the **TemporarySearch** element is used in the **Report** command, the search specifications used may allow the use of global variable references or may use rules such as [.words] and [.phrases] that preclude use of global variable references.

  Example:

  ```
  <SetVariable ConnectionId="1" Name="STAT" Value="NEW"/>
  <Report Id="report1" DatabaseId="DOCS" PageLayout="STARXML"
  ReportName="PLAIN">
        <SearchText>STS=$STAT AND KW=PRODUCTS</SearchText>
  </Report>
  ```

- in the search expression within a **SearchStatement** element of the **Search** command, to use the value of the global variable within the search.

  Example:

  ```
  <SetVariable ConnectionId="1" Name="MINDATE" Value="2000"/>
  <SetVariable ConnectionId="1" Name="MAXDATE" Value="2005"/>
  <Search DatabaseId="DOCS" Clear="Yes">
        <SearchStatement SearchStatementNumber="1">
              DEPT=PUBLICITY
        </SearchStatement>
        <SearchStatement SearchStatementNumber="2">
              PDATE=$MINDATE:$MAXDATE
        </SearchStatement>
        <SearchStatement SearchStatementNumber="3">
              S1 AND S2
        </SearchStatement>
  </Search>
  ```

- in a search expression within the **SearchLine** element of the **Search** command, to use the value of the global variable within the search. Note that the search specifications used may allow the use of global variable references in particular search lines or may use rules such as **[.Words]** and **[.Phrases]** that preclude use of global variable references.

  Example:

```
<SetVariable ConnectionId="1" Name="IDNO" Value="1234"/>
<Search DatabaseId="DOCS" Clear="Yes">
      <SearchLine SearchLineNumber="1">
            $IDNO
      </SearchLine>
      <SearchLine SearchLineNumber="2">
            BOOK; PERIODICAL; COLLECTION
      </SearchLine>
</Search>
```

- within the **SearchText** element of the **ShowIndex** command, to use the value of the global variable as the index term or pattern.

  Example:

```
<SetVariable ConnectionId="1" Name="AUTHOR" Value="PHILLIPS"/>
<ShowIndex DatabaseId="DOCS" Direction="Ascending" Case="upper"
LineCount="30" SearchPrefix="/BI">
      <SearchText>$AUTHOR</SearchText>
</ShowIndex>
```

- within the search expression assigned to the **Search Set** field within a **Field** element of the **SubmitGlobal** command, to use the value of the global variable within the search. Exception: Global variable references cannot be used within the search expression if the expression uses the **{DEFER}** feature, because it postpones the search execution until the time that the global is actually executed.

  Example:

```
<SetVariable ConnectionId="1" Name="OLD" Value="trash collector"/>
<SetVariable ConnectionId="1" Name="NEW" Value="materials engineer"/>
<SubmitGlobal ConnectionId="1">
      <Field InputFieldName="FLDOP">RT JOB "$OLD" "$NEW"</Field>
</SubmitGlobal>
```

3. Using STAR XML for Searching and Reporting

Perhaps the most important feature of STAR XML is the ability to search any STAR database and generate any STAR report, using the **Search**, **Report**, and **GetRecords** commands.

For search and report commands, and for the related **ShowIndex** command, database groups are treated as a single database: search results are dynamically summed, reports are dynamically merged on the sort key, and indices are dynamically merged alphabetically.

3.1 Searching

A single search can be applied to a single database or to multiple databases as a group. To apply a search to a single database, a STAR XML client would specify the database's **DatabaseId** in the **Search** command. To apply a search to multiple databases, a STAR XML client program would define a **DatabaseGroup** and add databases to it using the **Group** command, and then specify the **DatabaseGroupID** in the **Search** command.

STAR supports two types of searches: the **direct search** and the **filtered search**. They differ in how they are processed on the STAR host system. Direct searches are executed directly against a specified database. Filtered searches are interpreted using **search specifications**, which tell STAR how to combine pieces of a fill-in-the-blank search to create a final search. Direct searches provide searching freedom, while filtered searches provide both convenience and security.

For a given **Database** resource, only one type of searching can be used. If the database will be used for direct searching, the name of the database is specified. If the database will be used for filtered searching, the name of the **SERVER** record is specified and the actual database to be searched is stored in the **SERVER** record. This indirection is one of the features that provides both convenience and security, i.e., a STAR XML client program (1) does not need to specify the database name (convenience) and (2) cannot specify the database name (security).

If a STAR XML client program wants to search a database using both types of searches, it should define two **Database** resources, one for each type.

When a **DatabaseGroup** is defined, all databases in the group must use the same type of searches. If a STAR XML client program wants to search a group of databases using both types of searches, it should define two **DatabaseGroup** resources, one for each type, and add the same set of databases to each.

3.1.1 Direct Searches

Direct searches consist of a set of numbered **search statements**. Each search statement is a **STAR search expression**, meaning that it uses the syntax of STAR's search language.

Search statements may include single and multiword search terms, Boolean operators, proximity operators, wildcards, quotation marks, field qualifiers, ranging, parentheses, and all of STAR's other search syntax. Search statements may also contain references to other search statements, allowing you to combine results.

Search statements are associated with search statement numbers using the **SearchStatementNumber** attribute, which should have integer values, e.g.,

```
<SearchStatement SearchStatementNumber="1">
```

When these search statement numbers are to be referenced within search expressions, however, they are referred to with a leading **S** (*see* search statement number 3 in Example #2 below).

Direct searches are executed directly against the target database(s) in the order of their search statement numbers. By default, STAR clears all previous search statements and search results for the database or database group. When you use the attribute setting **Clear="False"**, STAR adds to or replaces any previous search statements for the database or database group, executes them, and also executes any previously specified search statements that have higher search statement numbers than any of those specified, in case they have dependencies on those just executed. In Example #3 below, search statement 2 is respecified, and STAR re-executes search statement 3 as well.

Example #1: One-search-statement search:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
      <Login>
            <Connection Id="M" Host="mars" UserId="sarah"/>
      </Login>
      <Database Id="newdocs" ConnectionId="M" DatabaseName="NEWDOCS"/>
      <Search DatabaseId="newdocs">
            <SearchStatement SearchStatementNumber="1">
                  YEAR=2002 AND KW=ANNOUNCEMENT AND NOT DEPT=HR
            </SearchStatement>
      </Search>
      <Quit/>
</Root>
```

Example #2: Multi-search-statement search:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
	<Login>
		<Connection Id="M" Host="mars" UserId="sarah"/>
	</Login>
	<Database Id="newdocs" ConnectionId="M" DatabaseName="NEWDOCS"/>
	<Search DatabaseId="newdocs">
		<SearchStatement SearchStatementNumber="1">
			STS=CURRENT
		</SearchStatement>
		<SearchStatement SearchStatementNumber="2">
			KW=PRODUCTS
		</SearchStatement>
		<SearchStatement SearchStatementNumber="3">
			(S1 OR S2) AND YEAR=2000
		</SearchStatement>
	</Search>
</Root>
```

Example #3: Continuation of Example #2 (which did not end with a **Quit** command):

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
	<Search DatabaseId="newdocs" Clear="False">
		<SearchStatement SearchStatementNumber="2">
			KW=PRODUCTS OR SERVICES
		</SearchStatement>
	</Search>
	<Quit/>
</Root>
```

Example #4: Searching a database group:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
<Login>
	<Connection Id="con" Host="jupiter" UserId="sarah"/>
</Login>
<Database Id="st" ConnectionId="con" DatabaseName="STOCK"/>
<Database Id="bk" ConnectionId="con" DatabaseName="BACKORDER"/>
<DatabaseGroup Id="group">
	<Add DatabaseId="st"/>
	<Add DatabaseId="bk"/>
</DatabaseGroup>
<Search DatabaseGroupId="group" Clear="Yes">
	<SearchStatement SearchStatementNumber="1">
		CUSTNO=0004239
	</SearchStatement>
</Search>
<Report Id="rpt1" DatabaseGroupId="group" PageLayout="STARXML"
ReportName="INVOICE">
	<SearchText>S1</SearchText>
</Report>
	<Quit/>
</Root>
```

3.1.2 Filtered Searches

Filtered searches consist of a set of numbered **search lines**. Each search line is a fragment of a search containing a search term, list of search terms, or a search expression.

To execute a filtered search, STAR XML requires a set of **search specifications** which are specified in a record in a **SERVER** database. The search specifications refer to the search lines by their search line numbers and can select, reformat, and combine the text in the search lines to create the set of search statements. The resulting search statements are executed by STAR as if they were from a direct search.

How the search lines are used to form a search is up to the designer of the search specifications. A given search line may be used for a single search term, a list of search terms, or a STAR search expression. Filtered searches and their search specifications may be used to provide convenience or security.

Example of convenience: A "what's new?" search uses a single search line for topic words. The search specifications search the words across a number of fields in records having dates within a certain number of days of the current date.

Example of security: A "department-restricted" search allows the any search expression to be used, but the search specifications limit the search to records having a department code that matches the one set in the user's **USERS** database record.

Filtered searches are executed in the order determined by the search specifications. The order of search lines in the request does not matter. By default, STAR clears all previous search lines and search results for the database or database group. When you use the attribute setting **Clear="False"**, STAR adds to or replaces any previous search lines for the database or database group and then re-executes the entire search specification, as in Example #2 below.

When you use a filtered search, you must be familiar with the particular set of search specifications you are using in order to know how many search lines to use, what type of data in what syntax should be supplied for each numbered search line, and what search statement number applies to the overall result (e.g., **S4**). When using a filtered search with a database group, you should make sure that the search specifications used for each database are identical or, at a minimum, agree with each other in these characteristics.

Search specifications are described in the Search Specifications section of the **STAR Reference Manual**.

Example #1: Filtered search:

```
<Search DatabaseId="newdocs">
       <SearchLine SearchLineNumber="1">SMITH*</SearchLine>
       <SearchLine SearchLineNumber="2">ENGINEERING;RESEARCH</SearchLine>
       <SearchLine SearchLineNumber="3">2000:2001</SearchLine>
</Search>
```

Example #2: Continuation of Example #1:

```
<Search DatabaseId="newdocs" Clear="False">
       <SearchLine SearchLineNumber="3">2002:2009</SearchLine>
</Search>
```

3.2 Reporting

Retrieving data from STAR records is a three-step process: (1) search, (2) generate a report, and (3) obtain records from the report.

The **Search** command performs a search against a database or database group. The result contains a **RecordCount** attribute specifying the number of STAR records retrieved. No report is generated and the records themselves are not returned.

The **Report** command generates a particular STAR report, given a search or search result, the name of the report, and other parameters. The result contains a **LineCount** attribute, specifying the number of lines in the report, and a **RecordCount** attribute, specifying the number of records in the report. Although the report is generated by STAR, the report content is not returned.

The number of records in a report usually matches the number retrieved by the search, but this is not the case when certain types of sorting are used. For example, a report sorted by a multiply occurring **Author** field will have a number of records that depends on the number of authors in the retrieved documents, not the number of documents.

Reports are always generated from a particular STAR search result. There are four ways in which the set of retrieved records can be specified:

1. A self-contained **Report** command can specify a search that will retrieve the records, e.g.:

```
<Report Id="report1" DatabaseId="NEWDOCS" PageLayout="STARXML"
ReportName="EXPORT">
     <SearchText>STS=CURRENT AND KW=PRODUCTS</SearchText>
</Report>
```

2. The results of a previously executed direct search can be referenced by one of its search statement numbers, e.g.:

```
<Search DatabaseId="newdocs">
     <SearchStatement SearchStatementNumber="1">
          STS=CURRENT
     </SearchStatement>
     <SearchStatement SearchStatementNumber="2">
          KW=PRODUCTS
     </SearchStatement>
     <SearchStatement SearchStatementNumber="3">
          S1:S2 AND YEAR=2000
     </SearchStatement>
</Search>
<Report Id="report1" DatabaseId="newdocs" PageLayout="STARXML"
ReportName="EXPORT">
     <SearchText>S3</SearchText>
</Report>
```

3. The results of a previously executed filtered search can be referenced by its final search statement number, e.g.:

```
<Search DatabaseId="newdocs">
     <SearchLine SearchLineNumber="1">CURRENT</SearchLine>
     <SearchLine SearchLineNumber="2">PRODUCTS</SearchLine>
     <SearchLine SearchLineNumber="3">2000</SearchLine>
</Search>
<Report Id="report1" DatabaseId="newdocs" PageLayout="STARXML"
ReportName="EXPORT">
     <SearchText>S6</SearchText>
</Report>
```

   In this case, you must know the appropriate search statement number (**S6** in this case) to use for the results of the particular search specifications you are using.

4. The results of a search using STAR's named search facility, where a search specification that uses a single search line is referenced, e.g.:

```
<Report Id="report1" DatabaseId="newdocs" PageLayout="STARXML"
ReportName="EXPORT">
     <SearchText>EVENTS; NEWS; ANNOUNCEMENTS</SearchText>
     <TemporarySearch ServerRecordName="SRCHDOCS" SearchName="KEYWORDS"/>
</Report>
```

The **GetRecords** command retrieves lines of a report, given a maximum number of records and the starting record number within the set of records in the report. The result can be quite large, depending on the number of records specified, the number of database fields in the report, and the sizes of the data in those fields.

When formatting STAR data in XML response documents, STAR XML can either format the data exactly as it appears in STAR (the default) or STAR XML can insert linebreaks and use indenting to make the document more suitable for display or printing. To allow STAR XML to use linebreaks and indenting, specify **`PrettyPrint="True"`** on the **GetRecords** command.

Example: Search and report:

Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
	<Login><Connection Id="con" Host="jupiter" UserId="sarah"/></Login>
	<Database Id="db" ConnectionId="con" DatabaseName="DOCS"/>
	<Search DatabaseId="db" Clear="Yes">
		<SearchStatement SearchStatementNumber="1">
			AN=EJ265940:EJ265941
		</SearchStatement>
	</Search>
	<Report Id="rpt1" DatabaseId="db" PageLayout="STARXML"
	ReportName="BROWSE">
		<SearchText>S1</SearchText>
	</Report>
	<GetRecords ReportId="rpt1" FirstRecord="1" RecordCount="10"
	PrettyPrint="True"/>
	<Quit/>
</Root>
```

Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<Root ecount="0">
	<Login><Connection Id="con" Host="jupiter" UserId="sarah"/></Login>
	<Database Id="db" ConnectionId="con" DatabaseName="DOCS"/>
	<Search DatabaseId="db" Clear="Yes" RecordCount="2">
		<SearchStatement SearchStatementNumber="1" RecordCount="2">
			AN=EJ265940:EJ265941
		</SearchStatement>
	</Search>
	<Report Id="rpt1" DatabaseId="db" PageLayout="STARXML"
	ReportName="BROWSE" RecordCount="2" LineCount="9">
		<SearchText>S1</SearchText>
	</Report>
	<GetRecords ReportId="rpt1" FirstRecord="1" RecordCount="10"/>
	<Response>
		<Record DatabaseId="db" RecordNumber="1000">
			<Field OutputFieldName="Sort">46585B4E48524A</Field>
			<Field OutputFieldName="TI">Reading Comprehension</Field>
			<Field OutputFieldName="AU">Tom Jenson</Field>
			<Field OutputFieldName="KWS">reading; comprehension;
			reading comprehension; understanding; books; knowledge;
			speed reading; vocabulary</Field>
		</Record>
		<Record DatabaseId="db" RecordNumber="1001">
			<Field OutputFieldName="Sort">4658585B4E48524A</Field>
			<Field OutputFieldName="TI">Teacher
			Responsibilities</Field>
			<Field OutputFieldName="AU">Rick Roberts</Field>
			<Field OutputFieldName="KWS">teachers; school; education;
			supervision; role models; responsibility; job
			assignments</Field>
		</Record>
	</Response>
	<Quit/>
</Root>
```

The special output field named **Sort** (distinguished by its upper/lowercase name) will always be included among the fields in a report. The data in this automatically generated field is derived from the sort key used for the report by STAR, and is used by STAR XML to merge reports from different databases when the report is generated from a database group. STAR XML Client programs will generally ignore the **Sort** output field.

4. Using STAR XML for Creating, Updating, and Deleting Records

STAR XML can be used to update a database by creating, updating, and deleting individual records, and by submitting a request for global processing to be carried out on a set of records as a background task.

To allow simultaneous use of STAR for searching, reporting, and updating by multiple users, STAR uses locks to preserve the integrity of each database. STAR XML client programs must obtain **write locks** before creating or updating individual records in a database.

In contrast, the **read locks** necessary for STAR XML's search and report functions are handled automatically by STAR XML, so STAR XML client programs need not deal with read locks. Write locks for deleting records and for global operations are also handled automatically.

If an XML request document does not include the commands to free all locks that are obtained during a session, any remaining locks are freed automatically by STAR XML when the session (i.e., connection) ends.

4.1 Creating Records

Creating a record in a STAR database is a two-step process. First, the record is reserved with the **CreateLock** command, which reserves a record number in the target database and returns it in a **RecordNumber** attribute. Subsequently, the record's data is filled in using the **UpdateRecord** command, which frees the lock upon completion.

In an interactive setting (when a STAR XML client program is responding to real-time commands from a user), the user may change his or her mind about creating a record. In such a case, the **FreeLock** command can be used to cancel creation of the record and to free the lock.

4.2 Updating Records

Updating (changing the contents of) a record in a STAR database is a two-step process. First, the record is locked with the **RecordLock** command. Subsequently, the record's data is changed using the **UpdateRecord** command, which frees the lock upon completion.

When a record is updated, the sub-elements of the **UpdateRecord** element specify data for some or all database fields. The STAR XML client program can specify whether fields in the record that are not mentioned in the **UpdateRecord** command should be emptied or left as they are. When data is supplied for a multiply occurring field, the STAR XML client program can specify whether the new occurrences should replace or be added to occurrences already in the field.

In an interactive setting (when a STAR XML client program is responding to real-time commands from a user), the user may change his or her mind about modifying a record. In such a case, the **FreeLock** command can be used to free the lock without changing the record.

4.3 Deleting Records

Deleting a record from a STAR database is a one-step process, using the **DeleteRecord** command.

4.4 Submitting Global Operations

Global operations are performed by **CASPER**, STAR's background process manager. **CASPER** gets its tasks from a queue and processes each task in turn. The **SubmitGlobal** command is used to add a request to this queue. After processing a **SubmitGlobal** command, the STAR XML host will continue processing commands. Eventually, the STAR XML client will disconnect from the STAR XML host and the STAR XML host will disconnect from the STAR host. **CASPER**, working asynchronously, will eventually perform the requested operation. This may occur either during the STAR XML session or after it has finished.

For more details on STAR's global functions, *see **Managing STAR Databases***, Chapter 2.

The following is an example of a **SubmitGlobal** command for changing the word "franc" to the word "euro" in the **UNIT** field of all records in the **CURRENCY** database:

```
<SubmitGlobal ConnectionId="star">
       <Field InputFieldName="SUBF">Change Values</Field>
       <Field InputFieldName="SETSP">/R 1:9999999</Field>
       <Field InputFieldName="UPDB">CURRENCY</Field>
       <Field InputFieldName="FLDOP" Clear="True">
              RT UNIT "franc" "euro"
       </Field>
</SubmitGlobal>
```

Appendix A. Starting and Stopping STAR XML Server

Before any STAR XML client program can use STAR XML, the **STAR XML Server** program must be running on one or more STAR XML hosts.

Unix

To start STAR XML Server (specifically, the STAR XML Server daemon) on a STAR XML host, type the following command in a shell:

```
% starxmldirectory/xmlserver-start
```

If you want STAR XML Server to run on other than the default port number of **11100**, add the **-p** ("port number") switch:

```
% starxmldirectory/xmlserver-start -p nnnnn
```

where *nnnnn* is the STAR XML Server port number.

To stop STAR XML Server on a STAR XML host, type the following command in a shell:

```
% starxmldirectory/xmlserver-stop
```

while logged in either as superuser or as the same user who started the STAR XML Server.

To check the status of STAR XML Server on a STAR XML host, type the following command in a shell:

```
% starxmldirectory/xmlserver-status
```

If you want STAR XML Server to start and stop automatically when you start and stop the STAR System, put the command

```
starxmldirectory/xmlserver-start
```

(with any switches you need) in the **post-on** section of file *~star/config/onoff* and the command

```
starxmldirectory/xmlserver-stop
```

in the **pre-off** section of file *~star/config/onoff*. Then you will no longer need to use the **xmlserver-start** and **xmlserver-stop** commands manually.

Windows

STAR XML Server, in the form of the **Starxml** service, should start automatically each time the computer is booted, so you should not have to start it manually. You can start, stop, or check the status of the **Starxml** service using the **Services** control panel.

Setting the Java Memory Allocation

By default STAR XML Server for Unix allocates 64 megabytes as the maximum Java memory. If a larger maximum memory allocation is required (something you might learn in consultation with the STAR Support desk) you can specify the memory allocation on the command line or in file *starxmldirectory/config/starxml.opt*.

On the command line:

```
% starxmldirectory/xmlserver-start –mx 128m
```

In file *starxmldirectory/config/starxml.opt*:

```
MX=128m
```

The suffix "m" stands for megabytes. If you want Java to use its default maximum memory allocation (which depends on the operating system, number of processors, physical RAM, and other factors), specify a blank value on the command line:

```
% starxmldirectory/xmlserver-start –mx ''
```

or in file *starxmldirectory/config/starxml.opt*:

```
MX=
```

Troubleshooting

When you are investigating a problem with STAR XML, you can use special switches, or start STAR XML Server directly, rather than with a script or service, to see whether error messages are displayed and/or to generate a log file. In a shell or windowing environment, STAR XML Server can be run interactively with realtime message displays.

The Unix **xmlserver-start** command has a switch that can be used for debugging. Use the **-i** ("interactive") switch to start STAR XML Server in a shell session. In this mode, you can direct logging messages to your shell (see Appendix F) and watch them appear as STAR XML Server runs. Do not close the shell session without first stopping STAR XML Server, which you do in this case by pressing Control-C rather than by using the **xmlserver-stop** command. Example:

```
% xml/xmlserver-start –i
```

To start STAR XML Server manually under either Unix or Windows, first turn off STAR XML Server (*see above*). Then invoke **STARXMLServer.jar** by double-clicking the **STARXMLServer.jar** icon or typing the following commands in a shell window:

```
% cd starxmlprogramdirectory
```

(Unix, or Windows Korn shell)

```
% javapath/java –jar STARXMLServer.jar –p nnnnn
```

```
> cd starxmlprogramdirectory
> javapath\java –jar STARXMLServer.jar –p nnnnn
```
(Windows DOS shell)

where

starxmlprogramdirectory is the STAR XML program directory, e.g., **/home/STAR/xml** or **/usr/local/starxml** for Unix or **c:\star\xml** or **c:\star xml** for Windows.

javapath is the directory in which Java itself, in particular the **java** command, was installed. If this directory is already in your PATH, you do not need to specify the path on this **java** command.

nnnnn (optional) is the STAR XML Server port number. You can omit the "-p" switch and the port number if you are using the default port number of **11100**. If you are not using the default port number, you must use the command-line syntax rather than double-clicking the **STARXMLServer.jar** icon.

Example on a STAR/Unix host:

```
% cd /home/STAR/xml
% /usr/local/java/bin/java –jar STARXMLServer.jar
```

To run STAR XML Server in a windowing environment, such as under Microsoft Windows, Solaris OpenWindows, CDE, or KDE:

Step 1: In a shell in a windowing environment, type the following commands:

```
% cd starxmlprogramdirectory
% javapath/java –jar STARXMLServer.jar -w –p nnnnn
```
(Unix, or Windows Korn shell)

```
> cd starxmlprogramdirectory
> javapath\java –jar STARXMLServer.jar -w –p nnnnn
```
(Windows DOS shell)

Step 2: A **STAR XML Server** window should appear. If not, check your shell window for either of these messages:

```
Can't connect to X11 window server
```

This message appears if you are not running in a windowing environment.

```
Exception in thread "main" java.lan.NoClassDefFoundError: STARXMLServer/jar
```

This message appears if you omitted the **-jar** switch.

Step 3: In the **STAR XML Server** window, select menu choice **File->Start Service**.

You can then use STAR XML client programs and observe the messages in the STAR XML Server window.

To stop STAR XML Server when you are done testing:

Step 1: In the **STAR XML Server** window, select menu choice **File->Stop Service**.

Step 2: Select menu choice **File->Exit**.

Appendix B. Using STAR XML Client

STAR XML Client is a simple Java-based testing program for STAR XML. It can be used on any computer, Unix or Windows, that has the Java 2 Runtime Environment. It can be used to test STAR XML by submitting XML request documents and observing the results. You must be using a windowing system, such as Microsoft Windows, Solaris OpenWindows, CDE, or KDE to use STAR XML Client.

STAR XML Client is delivered as file **STARXMLClient.jar**. To use it:

Step 1: Launch **STARXMLClient.jar**: Double-click the **STARXMLClient.jar** icon or type the following command in a shell window:

```
% cd starxmlprogramdirectory
% /javapath/java –jar STARXMLClient.jar
```

(Unix, or Windows Korn shell)

```
> cd starxmlprogramdirectory
> \javapath\java –jar STARXMLClient.jar
```

(Windows DOS shell)

where

starxmlprogramdirectory is the STAR XML program directory, e.g., **/home/STAR/xml** or **/usr/local/starxml** for Unix or **c:\star\xml** or **c:\star xml** for Windows.

javapath is the directory in which Java itself, in particular the **java** command, was installed. If this directory is already in your PATH, you do not need to specify the path on this **java** command.

A **STAR XML Client** window should appear.

Step 2: In the **STAR XML Client** window, fill in the **XML Server Host** and **XML Server Port** textboxes with the STAR XML host's hostname (e.g., **myxmlserver**) and port number (typically the default of **11000**). If STAR XML Server is running on the same computer as STAR XML Client, you can leave the **XML Server Host** value set to "localhost".

Step 3: Select menu choice **File->Open** to connect to the STAR XML host.

Step 4: Select menu choice **File->Get Request** to open an XML request document from the disk, which you can edit if you like. You can also type your XML request document directly.

Step 5: Select menu choice **File->Process Request** to submit the request to the STAR XML host.

Step 6: View the result. You can save the XML response document to disk using the **File->Save Output** menu choice.

Step 7: Repeat Steps 4 through 6 as many times as you like. If your previous XML request document ended with a **Quit** command, repeat Step 3 as well.

Step 8: Select menu choice **File->Exit** to close the connection to the STAR XML host and quit **STAR XML Client**.

STAR XML Client can be used to test STAR XML, to experiment with STAR XML functions, and to submit "canned" (predefined) requests without having to write your own STAR XML client programs. It is similar in purpose to the **starxml** command (*see* Appendix C) but allows you to examine and edit requests and results in a graphical window.

If you have a maintenance agreement that covers STAR XML, then STAR XML Client can be used to create and test an XML request document to be submitted to the STAR Support desk with a question or bug report.

Appendix C. Using the starxml Command

C.1 Using starxml in a Shell

starxml is a simple shell-level testing program for STAR XML. It lets you use or test STAR XML by submitting an XML request document to produce an XML response document. You do not need to be using a windowing system to use the **starxml** command.

The **starxml** command uses the syntax of a standard Unix filter, where either explicitly named files or standard input and standard output can be used for the input and output files.

To use the **starxml** command, prepare an XML request document (e.g., **request.xml**), or use a program that produces an XML request document, and type one of the commands shown below, where **myhost** is the hostname of the STAR XML host (not the STAR host).

The **starxml** program resides in the STAR XML directory. Add this directory to your PATH so you don't have to specify a pathname when you use the **starxml** command.

To display the results on the screen, use one of the following command forms:

```
% starxml myhost request.xml
% starxml myhost <request.xml
% (any command that produces an XML request) | starxml myhost
```

To store results in file **response.xml**, use one of the following command forms:

```
% starxml myhost request.xml response.xml
% starxml myhost request.xml >response.xml
% starxml myhost <request.xml >response.xml
% (any command that produces an XML request) | starxml myhost >request.xml
% (any command that produces an XML request) | starxml myhost - request.xml
```

If the STAR XML host is using a port number other than the default of **11100**, specify a colon and a port number after the host name, e.g.:

```
% starxml myhost:12345 request.xml response.xml
```

If the output file may already exist, add the **-d** switch, e.g.:

```
% starxml -d myhost request.xml response.xml
```

to indicate that an existing output file can be overwritten. Otherwise, **starxml** displays an error message if the output file already exists.

The **starxml** command can be used to use and test STAR XML, to experiment with STAR XML functions, and to submit "canned" (predefined) requests without having to write your own STAR XML client programs. It is similar in purpose to the **STAR XML Client** program (*see* Appendix B) but does not require you to use a windowing system.

If you have a maintenance agreement that covers STAR XML, then the **starxml** command can be used to test an XML request document to be submitted to the STAR Support desk with a question or bug report.

C.2 Using starxml in Scripts or Programs

The **starxml** command can be used by your own scripts and programs on the STAR host to handle the communication with STAR XML, so you don't have to write your own TCP/IP-level code.

Example: The following Bourne shell script, named **anlookup**, uses the **starxml** command to do a simple STAR search:

```
#! /bin/sh
# anlookup: shell script to look up DOCSMODEL accession numbers.
# Usage: anlookup record-number

RECNO="$1"

XMLHOST=`uname -n`
STARHOST=`uname -n`
STARID="carl"
STARPW="secret"

cat >in.temp <<EOF
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Root>
<Login>
<Connection Id="c" Host="$STARHOST" UserId="$STARID" Password="$STARPW"/>
</Login>
<Database Id="d" ConnectionId="c" DatabaseName="DOCSMODEL"/>
<Report Id="r" DatabaseId="d" PageLayout="STARXML" ReportName="EXPORT">
<SearchText>R=$RECNO</SearchText>
<FieldOptions OutputFieldNameList="AN" SortList=""/>
</Report>
<GetRecords ReportId="r" FirstRecord="1"/>
<Quit/>
</Root>
EOF

starxml $XMLHOST in.temp out.temp
AN=`grep 'OutputFieldName="AN"' out.temp | cut -d\> -f2 | cut -d\< -f1`
echo "Accession number of DOCSMODEL record $RECNO is $AN."

rm in.temp out.temp
exit 0
```

For brevity, error checking has been omitted from the script above. Error checking should always be included in shell scripts intended for "production" use, i.e., use by other than the author.

The **anlookup** shell script would be used as follows:

```
% anlookup 10

Accession number of DOCSMODEL record 10 is EJ281429.
```

Appendix D. Sample STAR XML Documents

D.1 Sample Request Document

The following is a sample XML request document. This document can be found in file *starxmldirectory*/**sample_request.xml.**

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- sample_request.xml: sample STAR XML request document -->
<Root>
   <!-- Get STAR XML version number -->
   <GetSTARXMLInfo/>
   <!-- Connect to STAR -->
   <Login>
      <Connection Id="mycon" Host="localhost" UserId="user1" Password="user1"/>
   </Login>
   <GetConnectionInfo ConnectionId="mycon"/>
   <Database Id="docs" ConnectionId="mycon" DatabaseName="DOCSMODEL"/>
   <Database Id="types" ConnectionId="mycon" DatabaseName="DOCSTYPES"/>
   <!-- Search DOCSMODEL for a certain record by Smith -->
   <Search DatabaseId="docs" Clear="True">
      <SearchStatement SearchStatementNumber="1">
         AU=SMITH* AND SUB=LEARNING THEORIES
      </SearchStatement>
      <SearchStatement SearchStatementNumber="2">
         YEAR &gt;= 2000 01 01
      </SearchStatement>
      <SearchStatement SearchStatementNumber="3">
         S1 AND NOT S2
      </SearchStatement>
   </Search>
   <!-- Generate report on retrieved record -->
```

```
   <Report Id="rpt1" DatabaseId="docs" PageLayout="STARXML" ReportName="EXPORT">
      <SearchText>S3</SearchText>
      <FieldOptions OutputFieldNameList="AU TI" SortList=""/>
   </Report>
   <GetRecords ReportId="rpt1" FirstRecord="1" PrettyPrint="True"/>
   <!-- Generate sorted report of TRANS field in DOCSTYPES database -->
   <Report Id="rpt2" DatabaseId="types" PageLayout="STARXML" ReportName="*ALL">
      <SearchText>TRANS=P*</SearchText>
      <FieldOptions OutputFieldNameList="TRANS" SortList="TRANS"/>
   </Report>
   <GetRecords ReportId="rpt2" FirstRecord="1" RecordCount="10" PrettyPrint="True"/>
   <!-- Add new Author to record 3 in DOCSMODEL database -->
   <RecordLock Id="3" DatabaseId="docs" RecordNumber="3"/>
   <UpdateRecord LockId="3" Clear="False">
      <Field InputFieldName="AU" Clear="False">Robinson, Marcy</Field>
   </UpdateRecord>
   <!-- Check freespace in the MODELS Space -->
   <AnalyzeSpace ConnectionId="mycon" SpaceName="MODELS"/>
   <Quit/>
</Root>
```

D.2 Sample Response Document

The following is an XML response document that might be returned for the sample XML request document in Section D.1. This document can be found in file *starxmldirectory*/**sample_response.xml**.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- sample_response.xml: sample STAR XML response document -->
<Root ecount="0">
   <!-- Get STAR XML version number -->
   <GetSTARXMLInfo Version="1.0"/>
   <!-- Connect to STAR -->
   <Login>
      <Connection Host="localhost" Id="mycon" Password="user1" UserId="user1"/>
   </Login>
   <GetConnectionInfo CharSet="8859-1" ConnectionId="mycon" Version="3.8.32"/>
   <Database ConnectionId="mycon" DatabaseName="DOCSMODEL" Id="docs"/>
   <Database ConnectionId="mycon" DatabaseName="DOCSTYPES" Id="types"/>
   <!-- Search DOCSMODEL for a certain record by Smith -->
   <Search Clear="True" DatabaseId="docs" RecordCount="1">
      <SearchStatement RecordCount="1"
         SearchStatementNumber="1">
            AU=SMITH* AND SUB=LEARNING THEORIES
         </SearchStatement>
      <SearchStatement RecordCount="80" SearchStatementNumber="2">
         YEAR &gt;= 2000 01 01
      </SearchStatement>
      <SearchStatement RecordCount="1" SearchStatementNumber="3">
         S1 AND NOT S2
      </SearchStatement>
   </Search>
   <!-- Generate report on retrieved record -->
   <Report DatabaseId="docs" Id="rpt1" LineCount="6" RecordCount="1"
   PageLayout="STARXML" ReportName="EXPORT">
      <SearchText>S3</SearchText>
      <FieldOptions OutputFieldNameList="AU TI" SortList=""/>
   </Report>
   <GetRecords FirstRecord="1" ReportId="rpt1" PrettyPrint="True">
      <Response>
         <Record DatabaseId="docs" RecordNumber="7">
            <Field OutputFieldName="Sort">FFFFFFF8</Field>
            <Field OutputFieldName="AU">Pagliaro, Louis Anthony.</Field>
            <Field OutputFieldName="AU">Smith, Brandon B.</Field>
            <Field OutputFieldName="TI">CAI in Pharmacology: Student
```

```
                Academic Performance and Instructional Interactions
                Research and Theory:  Designing and Managing Instruction</Field>
            </Record>
         </Response>
      </GetRecords>
      <!-- Generate sorted report of TRANS field in DOCSTYPES database -->
      <Report DatabaseId="types" Id="rpt2" LineCount="15" RecordCount="5"
      PageLayout="STARXML" ReportName="*ALL">
         <SearchText>TRANS=P*</SearchText>
         <FieldOptions OutputFieldNameList="TRANS" SortList="TRANS"/>
      </Report>
      <GetRecords FirstRecord="1" RecordCount="10" ReportId="rpt2" PrettyPrint="True">
         <Response>
            <Record DatabaseId="types" RecordNumber="19">
               <Field OutputFieldName="Sort">5646564A58  </Field>
               <Field OutputFieldName="TRANS">Paper</Field>
            </Record>
            <Record DatabaseId="types" RecordNumber="11">
               <Field OutputFieldName="Sort">564D555B55  </Field>
               <Field OutputFieldName="TRANS">Photo</Field>
            </Record>
            <Record DatabaseId="types" RecordNumber="20">
               <Field OutputFieldName="Sort">56584A594A545B465B4E5554  </Field>
               <Field OutputFieldName="TRANS">Presentation</Field>
            </Record>
            <Record DatabaseId="types" RecordNumber="16">
               <Field OutputFieldName="Sort">56584A595920584A524A46594A  </Field>
               <Field OutputFieldName="TRANS">Press Release</Field>
            </Record>
            <Record DatabaseId="types" RecordNumber="5">
               <Field OutputFieldName="Sort">565C58484D46594A205558494A58  </Field>
               <Field OutputFieldName="TRANS">Purchase Order</Field>
            </Record>
         </Response>
      </GetRecords>
      <!-- Add new Author to record 3 in DOCSMODEL database -->
      <RecordLock Id="3" DatabaseId="docs" RecordNumber="3"/>
      <UpdateRecord LockId="3" Clear="False">
         <Field InputFieldName="AU" Clear="False">Robinson, Marcy</Field>
      </UpdateRecord>
      <!-- Check freespace in the MODELS Space -->
      <AnalyzeSpace ConnectionId="mycon" MaxSpaceNumber="8"
         SpaceAllocated="6.00" SpaceFree="1324" SpaceName="MODELS"
         SpaceUsed="4.46"/>
      <Quit/>
   </Root>
```

Appendix E. Troubleshooting Tips

Although STAR XML and its documentation are provided with STAR for no additional charge, the standard STAR maintenance agreement does not include support for use of STAR XML in your own programs. Support is available for an additional charge. If you need help with a STAR XML client program you have written or that you had written for you, contact the STAR Support desk for further information.

The following tips may allow you to help yourself when you have trouble with STAR XML. We suggest that you review the following tips before asking the STAR Support deskfor help with a STAR XML program.

- Check that Java is working properly by typing these commands in the Unix shell or Windows DOS shell:

```
% cd starxmlprogramdirectory
% /javapath/java –jar HelloWorld.jar
```
(Unix)

```
> cd "starxmlprogramdirectory"
> "\javapath\java" –jar HelloWorld.jar
```
(Windows DOS shell)

where *starxmlprogramdirectory* is the STAR XML program directory and *javapath* is the directory in which the **java** command resides.

If the Java Runtime Environment is installed and working properly, the message "Hello, world!" will be displayed.

- Check that STAR XML Server has been started (*see* Appendix A).
- Check that the port on which STAR XML Server is running matches the port number requested by the STAR XML client.
- Check that the port on which STAR/Server is running (specified in file *inetd.conf*) matches the port requested in each connection of the **Login** command.
- Use the **xmltest** command to test STAR XML connections (see steps below).
- Run STAR XML Server with the **-l** switch or in a windowing environment (*see* Appendix A). Observe how STAR XML Server handles connections from STAR XML client programs by reading the messages it displays in its log file or graphical window and the shell window from which it was invoked.
- To check how STAR/Server handles connections from STAR XML Server, review the *xxxxx*.serlog and *xxxxx*.starlog files in the **~star/log** directory on the STAR host, where *xxxxx* is the STAR User ID specified in the connection of the **Login** command.
- To check whether the internet daemon (e.g., **inetd**) on a given Unix host is hearing connections, stop the internet daemon process and run it interactively for testing. Example for Solaris:

```
# cd /etc/rc2.d
# ./S72inetsvc stop
# /usr/sbin/inetd -s -d
```

- Check that the STAR User ID and password used with each connection of the **Login** command are valid for the STAR system being referenced. Check that the **USERS** database record for that STAR User ID is defined to allow STAR XML access in the **Interface** field. If the **USERS** record has a $SERVER specification in the *Dollar Variable* field, check that the specified database exists; if not, check that the default **SERVER** database exists.
- If a program you wrote is exhibiting a problem using STAR XML, have the program output its request document to a file, then test the file manually using **STAR XML Client** (*see* Appendix B) or the **starxml** command (*see* Appendix C). This lets you confirm whether the problem is a result of STAR XML's processing or another aspect of your program.
- If a particular XML request document is exhibiting a problem:
  - Reproduce the problem using **STAR XML Client** or the **starxml** command.
  - Gradually reduce the set of STAR XML commands in the document to find the smallest document that produces the problem. This narrows down the scope of the problem and identifies the combination of commands that must be investigated.
  - Try the same STAR XML commands with a different search, different database, different report, or even a different STAR host, to learn whether the problem is
  - Review the relevant portions of the **STAR XML Developer Manual** and the **STAR XML Reference Manual.**

How to use **xmltest**

For Unix, these steps can be performed on the STAR XML host or on any other host (e.g., the STAR host) that is running the same version of Unix, i.e., both Solaris, both Linux, etc.

For Windows, these steps can be performed only on the STAR host (because the Korn shell is required) and only if the STAR XML host is also running Windows.

Step 1: In a Unix shell or the Windows Korn shell, change to the STAR XML directory and run the xmltest script:

```
% cd starxmldirectory
% ./xmltest xmlhost:nnnnn starhost:mmmmm starid starpw
```

where

starxmldirectory is the STAR XML directory on the STAR XML host. If this directory is not located on the machine you are logged into, you will have to be able to cd to this directory over the network, e.g., **`cd /mountpoint/path/xml`** for Unix or **`cd //xmlhost/sharename/path/xml`** for Windows.

xmlhost is the hostname of the STAR XML host. If other than the default port number of **11100** is being used, add a colon and the port number after the hostname, e.g., **myhost:12345**.

starhost is the hostname of the STAR host. If other than the default port number of **11002** is being used, add a colon and the port number after the hostname, e.g., **myhost:12345**.

starid is a STAR User ID defined in the **USERS** database of this STAR host, with the **STAR XML** interface enabled in the **Interface** field, as described in **Step 1** above.

starpw is the STAR password for the STAR User ID. If the STAR password is blank, you can omit *starpw* from the command line. Note that having a STAR XML-enabled User ID with a blank password is a STAR security risk.

Step 2: Check the **xmltest** display, which should look as follows.

```
STAR XML Server on xmlhost is active.
STAR XML Server version nnn.

Login to STAR on starhost successful.
STAR version xxx, level yyy.
```

If error messages are displayed, try again with the -x switch, which cause the STAR XML request document and the STAR XML response document to be displayed, e.g.,

```
% ./xmltest -x xmlhost:nnnnn starhost:mmmmm starid starpw
```

Appendix F. Managing STAR XML Logging

STAR XML reports its ongoing actions in a log file, which can be used for debugging.

Message logging uses the **Log4j** version 2 logging framework. Options, specifying whether to log and where to log, are set using a configuration file named **log4j2.xml**. By editing this file you can enable or disable logging dynamically, without turning the STAR XML daemon or STAR XML service off and back on. If you invoke STAR XML directly from a command shell, you can even direct log messages to the "console" (command shell or window).

The setup steps below can be used to configure STAR XML logging, although in most cases the default settings should be good choices.

F.1 Logging Setup

You can configure or review logging settings when you install or update STAR XML or anytime later. You can change the settings at any time, whether the STAR XML daemon or STAR XML service is on or off. If the STAR XML daemon or service is already on, it will change to the settings you specify.

Step 1: Edit configuration file

Edit file **~star/xml/config/log4j2.xml** using either an XML editor or a text editor. Perform Step 2 through Step 5, then save the file.

Step 2: Set configuration file checks

The amount of time between STAR XML's checks for configuration file changes is controlled by the **monitorInterval** setting. This is the maximum amount of time you'll need to wait before a configuration file change goes into effect.

monitorInterval is specified in seconds and has a value between 5 and 3600.

The default setting of **monitorInterval="30"** should be fine in most cases.

If you want to change the interval, edit the **monitorInterval** attribute of the <**Configuration**> element.

Step 3: Set message patterns

The log file message format is controlled by the **pattern** setting. There is one pattern for each destination you might log to

Each **pattern** consists of constant text and variables defined at https://logging.apache.org/log4php/docs/layouts/pattern.html.

Pattern tips:

- The variable **%m** is for the message itself and should always be included.
- All other variables are optional, although you'd normally want to include **%n** for a terminating linebreak as well.
- For a console log, you may not need to see dates and times, while for a disk log file you can include **%d** followed by a date/time pattern.

The default settings of **pattern="%d{HH:mm:ss.SSS} %-5level - %m%n"** should be fine in most cases.

If you want to change the log file message format, edit the content of the <**pattern**> elements in the <**Appenders**> section. You can define one pattern for console log messages and one pattern for disk file log messages.

Example:

```
pattern="%m%n"
```

Step 4: Set log file location

The name and directory for a disk log file are controlled by the **fileName** setting.

The **fileName** can be a constant string with a path and filename, but it can also use the string **${sys:logroot}** to represent the STAR XML logging directory (**~star/xml/log**).

The default setting of **fileName="${sys:logroot}/starxml.log"**, sets the log file to **~star/xml/log/starxml.log**, which should be fine in most cases.

If you want to change the log file name or location, edit the value of the **filename** attribute of the <**File name="LogToDefault"**> elements in the <**Appenders**> section.

Examples:

```
filename="${sys:logroot}/april.log"
filename="/home/logfiles/starxml.log"
```

Step 5: Set level of detail

The amount of information in the log file is controlled by the **level** setting.

The **level** can be set to **trace**, **info**, **warn**, **error**, or **fatal**, in decreasing order of details. Choose **trace** to log all possible messages, choose **fatal** to log only the most important messages, or make another choice along the scale.

The default setting of **level="trace"** should be fine in most cases.

If you want to change the **level** setting, edit the **level** attribute of the **<Root>** element in the **<Loggers>** section.

Example:

```
level="error"
```

For more details on **log4j** configuration, see http://logging.apache.org/log4j/2.x/manual/configuration.html.

F.2 Enabling STAR XML Logging to a Disk File

Step 1. Edit file **~star/xml/config/log4j2.xml** using either an XML editor or a text editor.

Step 2. In the **<Root>** section of the **<Loggers>** section, comment out the **AppenderRef** element for **LogToConsole** and comment in the **AppenderRef** element for **LogToDefault**.

```
<!--<AppenderRef ref="LogToConsole"/>-->
<AppenderRef ref="LogToDefault"/>
```

STAR XML will notice the change within 30 seconds (unless you changed the amount of time between configuration checks) and will start logging to the specified disk file.

F.3 Enabling STAR XML Logging to the Console

When you invoke STAR XML interactively, using the **xmlserver-start -i** command under Unix or the appropriate **java -jar** command under Windows, you can use the option to send STAR XML logging to the console, which will be your Unix shell session or Windows command window.

Step 1. Edit file **~star/xml/config/log4j2.xml** using either an XML editor or a text editor.

Step 2. In the **<Root>** section of the **<Loggers>** section, comment in the **AppenderRef** element for **LogToConsole** and comment out the **AppenderRef** element for **LogToDefault**.

```
<AppenderRef ref="LogToConsole"/>
<!--<AppenderRef ref="LogToDefault"/>-->
```

It may be simplest to make this change before you start STAR XML, but if it's already active then STAR XML will notice the change within 30 seconds (unless you changed the amount of time between configuration checks) and will start logging to the console.

F.4 Disabling STAR XML Logging

Step 1. Edit file **~star/xml/config/log4j2.xml** using either an XML editor or a text editor.

Step 2. In the **<Root>** section of the **<Loggers>** section, comment out the **AppenderRef** element for **LogToConsole** and the **AppenderRef** element for **LogToDefault**.

```
<!--<AppenderRef ref="LogToConsole"/>-->
<!--<AppenderRef ref="LogToDefault"/>-->
```

STAR XML will notice the change within 30 seconds (unless you changed the amount of time between configuration checks) and will stop logging.