AXL Troubleshooting

This chapter contains the following topics:

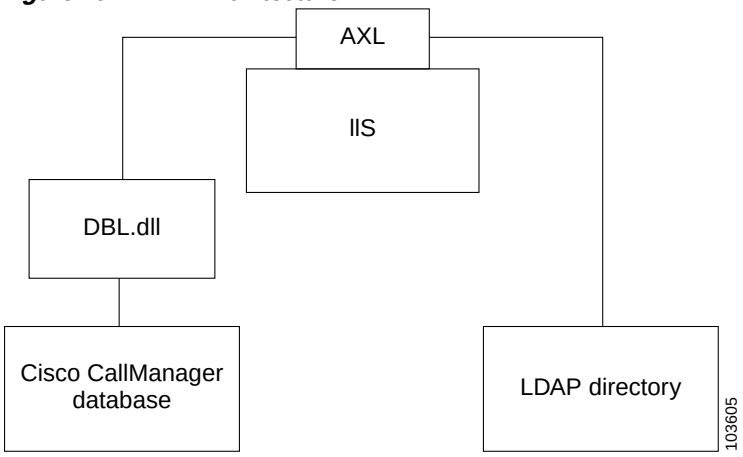
Overview

AVVID XML Layer (AXL) is a Cisco application programming interface (API) and web service designed to give applications access to Cisco CallManager configuration and provisioning services. AXL is implemented as a Simple Object Access Protocol (SOAP) over HTTP web service, in which requests in the form of extensible markup language (XML) documents are sent from the application to the Cisco CallManager's web server, which responds with an XML-formatted response. Security is handled through the Cisco CallManager's Internet Information Services (IIS) security mechanism.

Architecture

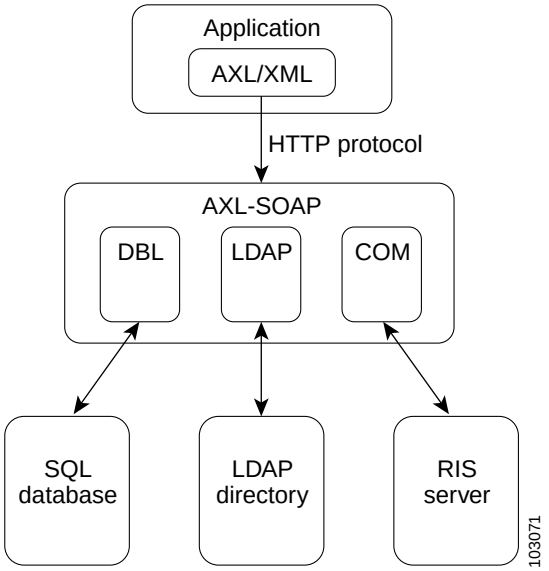
AXL is hosted as an IIS dynamic-link library or ISAPI.dll (SOAPISAPI.dll) as shown in Figure 20. AXL accesses either the Cisco CallManager configuration database, or the embedded DC-Directory Lightweight Directory Access Protocol (LDAP) directory to read or write configuration data. Database access is handled by the DBL.dll layer, which enforces the Cisco CallManager database business rules. This same DBL.dll layer serves the Cisco CallManager Administration program’s graphical user interface (GUI) pages.

Figure 20 AXL Architecture



The AXL API methods, known as requests, use a combination of HTTP and SOAP as shown in Figure 21. SOAP is an XML remote procedure call protocol. Users perform requests by sending XML data to the Cisco CallManager Real-Time Information Server (RIS). The server then returns the AXL response, which is also a SOAP message.

Figure 21 AXL Request Flow



Postinstallation Checklist

The AXL web service is enabled by default on Cisco CallManager servers and does not have to be installed or configured. Use the following checklist to avoid some common problems that may be avoided by fine-tuning your configuration before proceeding with the troubleshooting process:

____ If the AXL client application is unable to connect to the AXL service, check the following:

– The AXL application has been configured with the correct IP address of the AXL server.

- The AXL application has been configured with the appropriate AXL user credentials.
- The user credentials have the correct access rights, as described in the "Postinstallation Checklist" on page 36.
- The application server has HTTP connectivity to the AXL server (that is, port 80).
- HTTPS (secure) has been configured for AXL.

____ If the AXL functions or requests are failing, check the following:

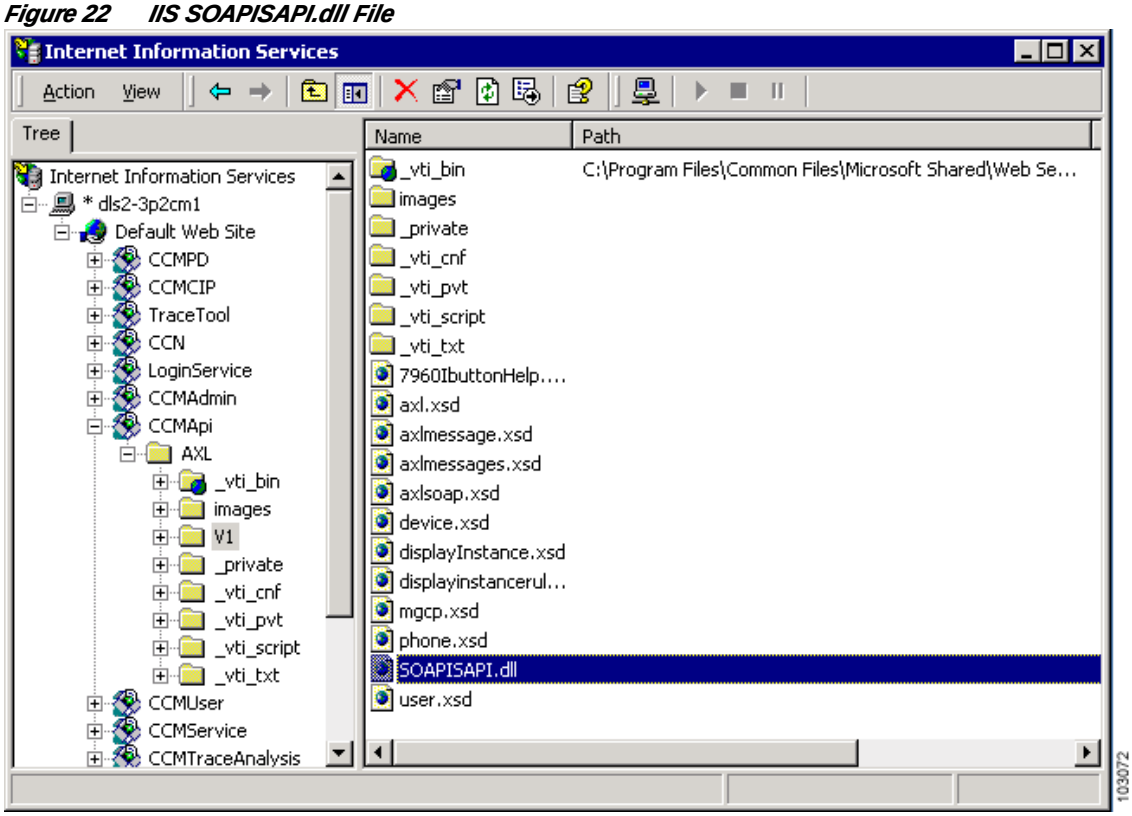
- AXL logs for AXL or SOAP error responses. See "Error Codes" on page 43.
- IIS logs for general web server errors or problems

Note DBL logs should be enabled only on request from Cisco TAC or Cisco Developer Services.

____ Check that users other than the Microsoft Windows Administrator group that require access to AXL have read and execute rights to the SOAPISAPI.dll file.

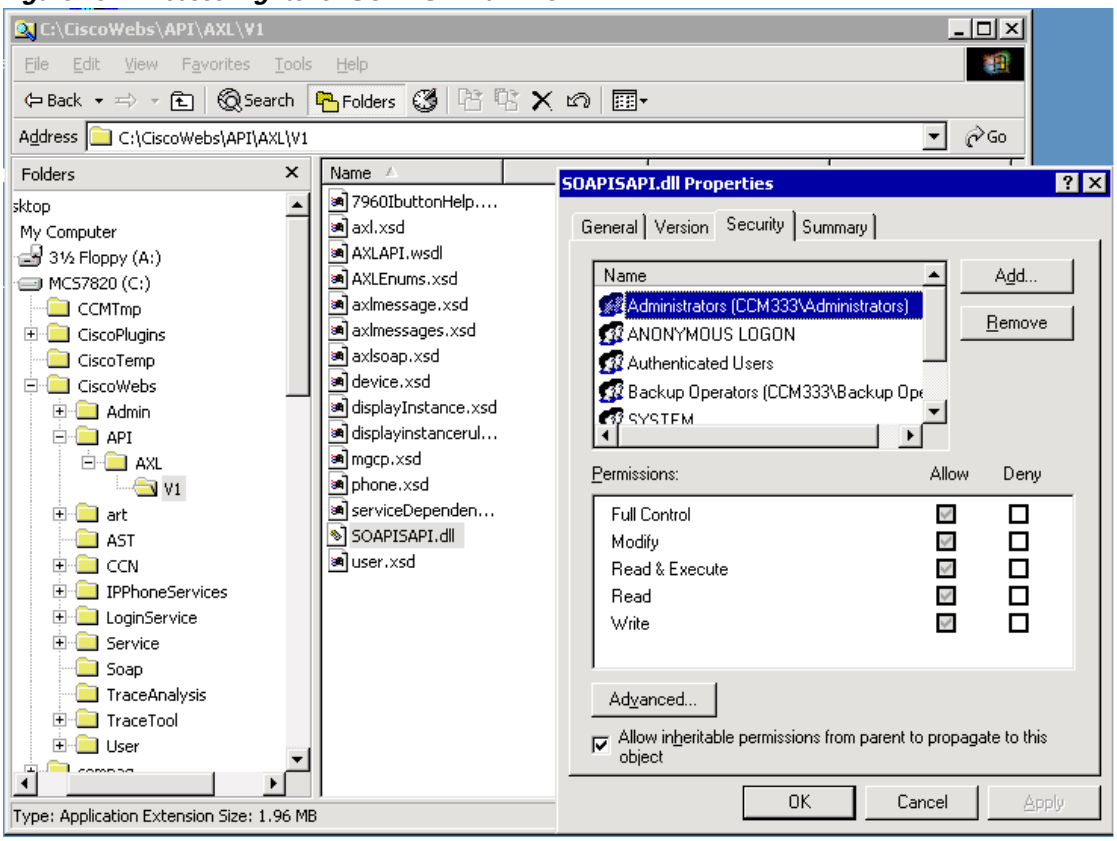
By default, only the Windows Administrator group is given access to the SOAPISAPI.dll file, restricting AXL access to the administrator. Figure 22 shows the location of the SOAPISAPI.dll file.

Figure 22 IIS SOAPISAPI.dll File



To enable AXL access to additional users, simply give the new users read and execute rights to the file C:\CiscoWebs\API\AXL\V1\SOAPISAPI.dll. Figure 23 shows the Properties window through which you can add and change access rights to a file.

Figure 23 Access Rights for SOAPISAPI.dll File



____ Check that applications in a cluster configuration are connected to the AXL service only on the Cisco CallManager Publisher server.

____ Verify basic AXL functionality by performing the procedure that follows:

Verifying AXL Functionality Procedure

Step 1 Open Notepad on the client PC by choosing **Start** > **Programs** > **Accessories** > **Notepad**.

Step 2 Copy the following HTML text into the untitled file:

```
<html>
<body>
<form method="POST" action="http://**CCM IP**/CCMAPI/AXL/V1/SOAPISAPI.dll">
  <input type="text" name="XML" size="20"><input type="submit">
</form>
</body>
</html>
```

Step 3 Change **CCM IP ** to the IP address of the AXL Cisco CallManager server.

Step 4 Save the file to the desktop as axltest.htm.

Step 5 Open the file with Internet Explorer.

Step 6 Click the **Submit Query** button.

Step 7 Enter the AXL access user credentials.

Step 8 Verify that the resulting page is similar to the following:

```
- <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
- <SOAP-ENV:Body>
- <SOAP-ENV:Fault>
  <faultcode>SOAP-ENV:Client</faultcode>
  <faultstring>The AXL API service can only accept Content-Type: text/xml</faultstring>
  </SOAP-ENV:Fault>
  </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

The resulting page confirms that the client PC has HTTP connectivity to the AXL server, that the AXL user credentials are valid, and that the AXL service is running.

____ Check the Cisco Database Layer service parameter MaxAXLWritesPerMinute:

Updating the Cisco CallManager database can have the side effect of adversely affecting system performance. To prevent this effect, the system administrator can control how many AXL requests are allowed to update the database per minute through the Database Layer service parameter MaxAXLWritesPerMinute.

AXL accommodates all requests until the MaxAXLWritesPerMinute value is reached. The default value of this parameter is 20, and the maximum configurable value is 999. However, the maximum Cisco supported value for production systems is 60.

After the MaxAXLWritesPerMinute value is reached, attempts to modify the database with AXL are rejected with an HTTP 503 Service Unavailable response. Every minute, AXL resets its internal counter and begins to accept AXL update requests until the MaxAXLWritesPerMinute value is reached.

Troubleshooting Tools

This section describes available troubleshooting tools.

AXL Trace Logs

AXL trace logs contain the text of every AXL request and response, along with user and origination IP information. Trace logs are useful for identifying who is making AXL requests, inspecting the AXL XML request for format or syntax errors, and determining the actual AXL service's response or errors.

AXL trace logs are created in the C:\Program Files\Cisco\Trace\AXL directory. AXL creates up to 50 1-MB text files in this directory. Once 50 files have been written, the oldest file is overwritten by the newest file.

Note Restarting the Cisco CallManager's WWW service (IIS), either manually or with a machine reset, causes trace logs to begin overwriting the oldest file.

The AXL trace log contains:

- Time that the request was received
- Client IP address
- Client user ID
- Request ID number

- Request contents
- Response contents

The following is sample AXL trace log output:

```
RequestTime: 01/15/2003-11:24:49
Client IP: 10.101.156.108
Client User ID: user1
RequestID: 1
RequestContents:
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <SOAP-ENV:Body> <m:getPhone
xmlns:m="http://www.cisco.com/AXL/API/1.0" sequence="1"> <phoneName>Joe_Phone</phoneName>
<!-- <phoneId>{2FC4F3C0-FC99-4646-A44A-137356B6289C}</phoneId>  --> </m:getPhone>
</SOAP-ENV:Body> </SOAP-ENV:Envelope>
ResponseContents:
<SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"> <SOAP-ENV:Body>
<SOAP-ENV:Fault> <faultcode>SOAP-ENV:Client</faultcode> <faultstring> <![CDATA[Item not
valid: The specified phone was not found.]]></faultstring> <detail
xmlns:axl="http://www.cisco.com/AXL/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.cisco.com/AXL/1.0
http://user1-w2k/CCMApi/AXL/V1/axlsoap.xsd">
<axl:error  sequence="1"> <code>2</code> <message> <![CDATA[Item not valid: The specified
phone was not found.]]></message> <request>getPhone</request> </axl:error> </detail>
</SOAP-ENV:Fault> </SOAP-ENV:Body> </SOAP-ENV:Envelope>
```

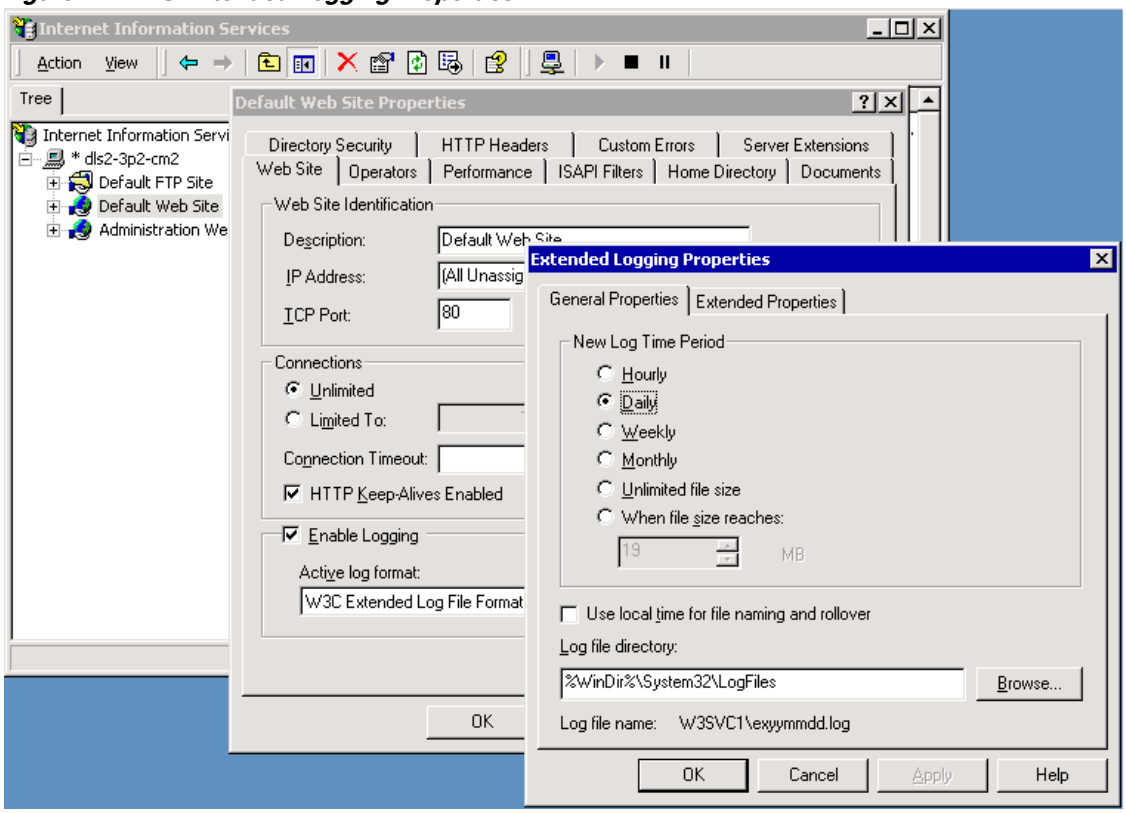
IIS Logs

IIS logs record user sessions with the IIS web service. Information recorded is:

- Username
- IP address
- Access method
- Uniform resource identifier (URI) requested
- Potentially other detailed information about the session and client

IIS logs are configured from the IIS manager. Only a few of the possible IIS logging information fields are enabled by default. Refer to the Microsoft IIS documentation for more information about the additional logging options shown in Figure 24.

Figure 24 IIS Extended Logging Properties



Note IIS serves many functions on Cisco CallManager. Not all IIS log entries are related to AXL access.

DBL Logs

Database Layer (DBL) logs document the interaction between the AXL web service and the supporting DBL and Structured Query Language (SQL) database services. Typically these logs are necessary only when debugging a problem internal to AXL, DBL, or SQL.

Perform the following steps to enable DBL logs:

Step 1 From the Cisco CallManager Administration window, choose **Application > Cisco CallManager Serviceability**.

The Cisco CallManager Serviceability window shown in Figure 25 displays.

Step 2 Choose **Trace > Configuration**.

Step 3 From the Servers column, choose the server.

The server that you chose displays next to the Current Server title, and a box with configured services displays.

Step 4 From the Configured Services box, choose the **Cisco Database Layer Monitor** service.

The service that you chose displays next to the Current Service title, along with the current server that you chose. The trace parameters display for the service that you chose.

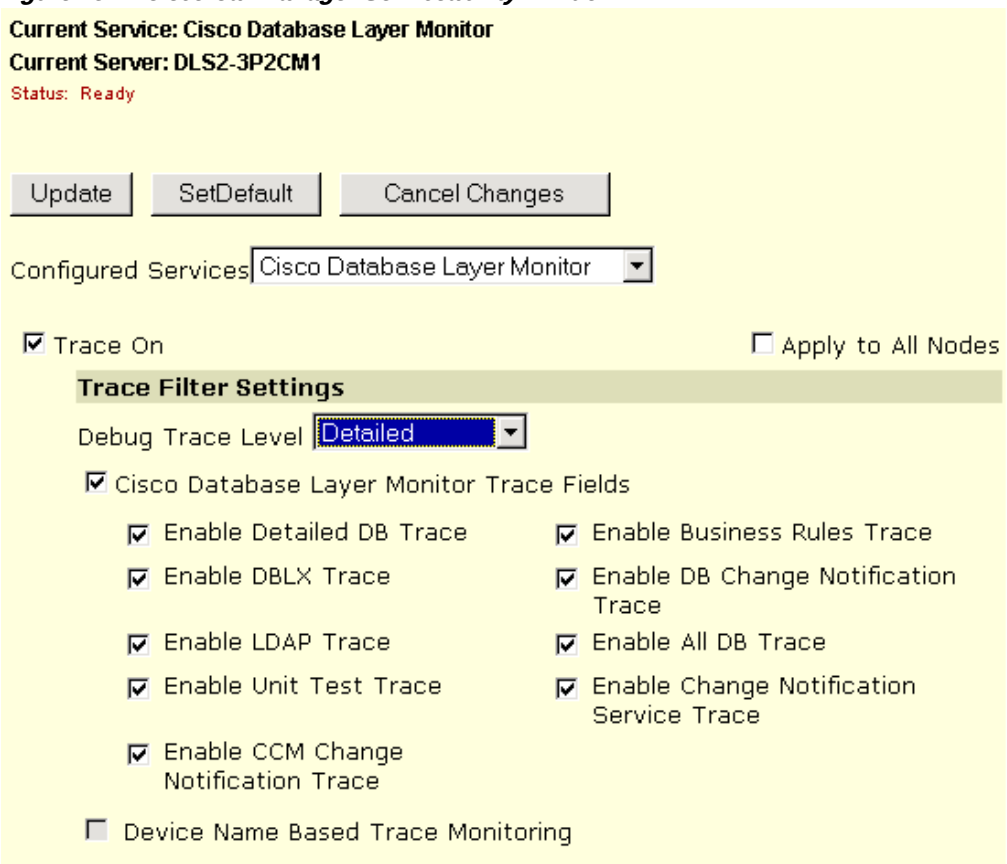
Note Only the trace parameters for the service that you chose display. The display shows all other parameters dimmed.

Step 5 Check the Trace On check box.

Step 6 If you want trace to apply to all Cisco CallManager servers in the cluster, check the Apply to All Nodes check box.

Step 7 In the Debug Trace Level field, click the drop-down arrow and click **Detailed**.

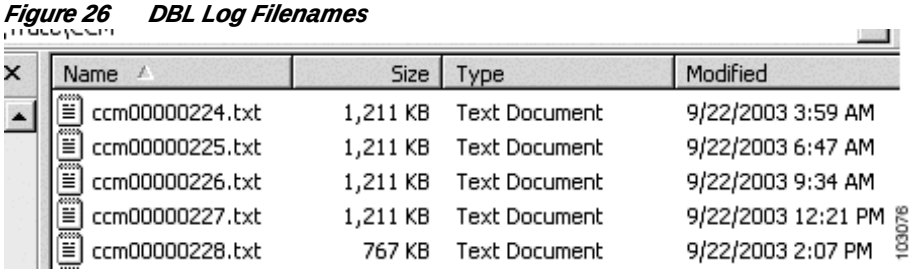
Figure 25 Cisco CallManager Serviceability Window



Note the output trace file path for later log collection.

The DBL Layer tracing mechanism uses circular logging, which means that log entries are written to a single log file until it reaches a certain configurable size. Once that size has been reached, subsequent log entries are written to a new log file with a sequentially incremented filename, as shown in Figure 26.

Figure 26 DBL Log Filenames



The number of log files available is also a limited configurable number. When the last log file is full, the logging process deletes the first log file and reuses its filename. File overwriting proceeds in a circular manner, with newer log files overwriting existing log files with the same name.

When you work with circular logging systems, it is important that you balance the disk space requirement, which is calculated by multiplying the maximum size of a log file by the maximum number of log files allowed, with the required log file collection interval. If the size and number of log files are too large, a large amount of disk space can be consumed by logging. If the size and number of log files are too small, log files can be overwritten before the administrator can collect them.

Error Codes

If an exception occurs on the server, or if any other error occurs during the processing of an AXL request, an error is returned in the form of a SOAP Fault message, such as in the following example:

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Client</faultcode>
            <faultstring>
            <![CDATA[
            An error occurred during parsing
            Message: End element was missing the character '>'.

            Source = Line : 41, Char : 6
            Code : c00ce55f, Source Text : </re
            ]]>
            </faultstring>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP Fault messages can also contain more detailed information. The following is an example of a detailed SOAP Fault:

```
<SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/" S
OAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <SOAP-ENV:Body>
        <SOAP-ENV:Fault>
            <faultcode>SOAP-ENV:Client</faultcode>
          <faultstring>Device not found with name SEP003094C39708.</faultstring>
            <detail  xmlns:axl="http://www.cisco.com/AXL/1.0"
                     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                     xsi:schemaLocation="http://www.cisco.com/AXL/1.0
                                         http://myhost/CCMApi/AXL/V1/axlsoap.xsd">
                <axl:error  sequence="1234">
                    <code>0</code>
                    <message>
```

```
<![CDATA[
Device not found with name SEP003094C39708.
]]>
                    </message>
                    <request>doDeviceLogin</request>
                </axl:error>
            </detail>
        </SOAP-ENV:Fault>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Error codes will be included in the <detail> element of a SOAP Fault. The errors are represented by the axl:Error element. If a response to a request contains an <error> element, the user agent can determine the cause of the error by looking at the subelements of the <error> tag.

The following list describes the <error> element.

- code—The <code> element is a numerical value that is used by the user agent to find out what type of error has occurred. The error codes are described in Table 3.

Table 3 Error Code Descriptions

Error Code	Description
Less than 5000	These are errors that directly correspond to DBL Exception error codes.
5000	Unknown Error—An unknown error occurred while processing the request. This error can be caused by a problem on the server, but it can also be caused by errors in the request.
5002	Unknown Request Error—This error occurs if the user agent submits a request that is unknown to the API.
5003	Invalid Value Exception—This error occurs if an invalid value is detected in the XML request.
5004	AXL Unavailable Exception—This error occurs if the AXL service is too busy to handle the request at that time. The request should be sent again at a later time.
5005	Unexpected Node Exception—This error occurs if the server encounters an unexpected element. For example, this error is returned if the server expects the next node to be <name>, but encounters <protocol>. These errors are always caused by malformed requests that do not adhere to the latest AXL schema.

- message—The <message> element is provided so that the user agent gets a detailed error message explaining the error.
- request—The <request> element is provided so that the user agent can determine what type of request generated this error. This element is optional; therefore it may not always appear.