CHAPTER 7

Cisco XML and Large Data Retrieval

XML for the router supports the retrieval of large XML responses in blocks (for example, chunks or sections).

These sections provide information about large data retrieval:

- Iterators, page 7-93
- Throttling, page 7-98
- Streaming, page 7-99

Iterators

When a client application makes a request, the resulting response data size is checked to determine whether it is larger than a predetermined block size.

If the response data is not larger than the predetermined block size, the complete data is returned in a normal response.

If the response data is larger than the block size, the first set of data is returned according to the block size along with a decremented iterator ID included as the value of the IteratorID attribute. The client must then send <GetNext> requests including the iterator ID until all data is retrieved. The client application knows that all data is retrieved when it receives a response that does not contain an IteratorID attribute.

Usage Guidelines

These points should be noted by the client application when iterators are used:

- The block size is a configurable value specific to each transport mechanism on the router; that is, the XML agent for the dedicated TCP connection and Secure Shell (SSH), Telnet, or Secure Sockets Layer (SSL) dedicated TCP connection.

  Use this command to configure the iteration size:

  xml agent [**tty** | **ssl**] **iteration on size** <*1-100000*>

  Specify the iteration size in KB. The default is 48 KB.

Note The **iteration** command includes the option to turn off the XML response iterator. However, we do not recommend turning off the iterator because of the large memory usage that occurs temporarily.

- The block size refers to the entire XML response, not just the payload portion of the response.
- Large responses are divided based on the requested block size, not the contents. However, each response is always a complete XML document.
- Requests containing multiple operations are treated as a single entity when the block size and IteratorID are applied. As a result, the IteratorID is an attribute of the <Response> tag, never of an individual operation.
- If the client application sends a request that includes an operation resulting in the need for an iterator to return all the response data, any further operations contained within that request are rejected. The rejected operations are resent in another request.
- The IteratorID is an unsigned 32-bit value that should be treated as opaque data by the client application. Furthermore, the client application should not assume that the IteratorID is constant between <GetNext> operations.

To reduce memory overhead and avoid memory starvation of the router, these limitations are placed on the number of allowed iterators:

- The maximum number of iterators allowed at any one time on a given client session is 10.
- The maximum number of iterators allowed at any one time for all client sessions is 100.

  If a <Get> request is issued that results in an iterated response, it is counted as one iterator, regardless of the number of <GetNext> operations required to retrieve all of the response data.

  For example, a <Get> request may require 10, 100, or more <GetNext> operations to retrieve all the associated data, but during this process only one iterator is being used.

  Also, an iterator is considered to be in use until all of the response data associated with that iterator (the original <Get> request) is retrieved or the iterator is terminated with the Abort attribute.

Examples Using Iterators to Retrieve Data

This example shows a client request that utilizes an iterator to retrieve all global Border Gateway Protocol (BGP) configuration data for a specified autonomous system:

Sample XML Client Request to Retrieve All BGP Configuration Data

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion=”35” MinorVersion=”2”>
        <AS>
          <Naming>
            <AS>0</AS>
          </Naming>
           <FourByteAS>
          <Naming><AS>3</AS></Naming>
          <DefaultVRF/>
         </FourByteAS>
        </AS>
      </BGP>
```

```
    </Configuration>
  </Get>
</Request>
```

Sample XML Response from the Router Containing the First Block of Retrieved Data

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0" IteratorID=”1”>
  <Get>
    <Configuration>
      <BGP MajorVersion=”24” MinorVersion=”0”>
        <AS>
          <Naming>
            <AS>0</AS>
          </Naming>
          <FourByteAS>
            <Naming><AS>3</AS></Naming>
            <DefaultVRF>
            ...
              1st block of data returned here
            ...
            </DefaultVRF>
          </FourByteAS>
        <AS>
      </BGP>
    </Configuration>
  <Get>
  <ResultSummary ErrorCount="0"/>
</Response>
```

Second XML Client Request Using the <GetNext> Iterator to Retrieve the Next Block of BGP Configuration Data

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <GetNext IteratorID=”1”/>
</Request>
```

Sample XML Response from the Router Containing the Second Block of Retrieved Data

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0" IteratorID=”1”>
  <Get>
    <Configuration>
      <BGP MajorVersion=”24” MinorVersion=”0”>
       <AS>
          <Naming>
            <AS>0</AS>
          </Naming>
          <FourByteAS>
            <Naming><AS>3</AS></Naming>
            <DefaultVRF
            ...
              2nd block of data returned here
            ...
            </DefaultVRF>
          </FourByteAS>
       </AS>
      </BGP>
    </Configuration>
  <Get>
  <ResultSummary ErrorCount="0"/>
</Response>
```

Third XML Client Request Using the <GetNext> Iterator to Retrieve the Next Block of BGP Configuration Data

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <GetNext IteratorID=”1”/>
</Request>
```

Sample XML Response from the Router Containing Third Block of Retrieved Data

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0" IteratorID=”1”>
  <Get>
    <Configuration>
      <BGP MajorVersion=”24” MinorVersion=”0”>
        <AS>
          <Naming>
            <AS>0</AS>
          </Naming>
          <FourByteAS>
            <Naming><AS>3</AS></Naming>
            <DefaultVRF>
            ...
              3rd block of data returned here
            ...
            </DefaultVRF>
          </FourByteAS>
        </AS>
      </BGP>
    </Configuration>
  </Get>
  <ResultSummary ErrorCount="0"/>
</Response>
```

Final XML Client Request Using the <GetNext> Iterator to Retrieve the Last Block of BGP Configuration Data

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <GetNext IteratorID=”1”/>
</Request>
```

Final XML Response from the Router Containing the Final Block of Retrieved Data

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion=”24” MinorVersion=”0”>
        <AS>
          <Naming>
            <AS>0</AS>
          </Naming>
          <FourByteAS>
            <Naming><AS>3</AS></Naming>
            <DefaultVRF>
            ...
              Final block of data returned here
            ...
            </DefaultVRF>
          </FourByteAS>
        </AS>
      </BGP>
    </Configuration>
  </Get>
  <ResultSummary ErrorCount="0"/>
</Response>
```

Large Response Division

The default behavior for large response division is that large responses are divided based on the requested block size.

To specify a different basis for the division, use the IterateAtFirstTableGet attribute in the <Get> tag.

Sample XML Request with attribute IterateAtFirstTable

```
<?xml version="1.0" encoding="UTF-8"?>
<Request>
 <Get IterateAtFirstTable="true">
    <Operational>
       <BGP>
         <Active>
           <DefaultVRF>
            <AFTable>
              <AF>
                <Naming>
                  <AFName Match="*"/>
                </Naming>
               <PathTable>
                 <Path>
                   <Naming>
                     <RD Match="*"/>
                     <Network Match="*"/>
                     <NeighborAddress Match="*"/>
                     <RouteType Match="*"/>
                     <SourceRD Match="*"/>
                   </Naming>
                 </Path>
                </PathTable>
               </AF>
            </AFTable>
          </DefaultVRF>
         </Active>
       </BGP>
    </Operational>
 </Get>
</Request>
```

Terminating an Iterator

A client application may terminate an iterator without retrieving all of the response data by including an Abort attribute with a value of “true” on the <GetNext> operation. A client application that does not complete or terminate its requests risks running out of iterators.

This example shows a client request using the Abort attribute to terminate an iterator:

Sample XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion=”24” MinorVersion=”0”>
        <AS>
          <Naming>
            <AS>0</AS>
          </Naming>
          <FourByteAS>
```

```
            <Naming><AS>3</AS></Naming>
          <DefaultVRF/>
          </FourByteAS>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Request>
```

Sample XML Response from the Router

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0" IteratorID=”2”>
  <Get>
    <Configuration>
      <BGP MajorVersion=”24” MinorVersion=”0”>
        <AS>
          <Naming>
            <AS>0</AS>
          </Naming>
          <FourByteAS>
            <Naming><AS>3</AS></Naming>
             <DefaultVRF>
            ...
              1st block of data returned here
            ...
             </DefaultVRF>
          </FourByteAS>
        <AS>
      </BGP>
    </Configuration>
  <Get>
  <ResultSummary ErrorCount="0"/>
</Response>
```

Sample XML Request Using the Abort Attribute to Terminate an Iterator

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <GetNext IteratorID=”2” Abort=”true”/>
</Request>
```

Sample XML Response from the Router

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <GetNext IteratorID=”2” Abort=”true”/>
</Response>
```

Throttling

XML response data could be large resulting in high CPU utilization or high memory usage when constructing the XML response. Throttling mechanisms in the XML agent provide a means for external users or an NMS to control the impact to the system.

CPU Throttle Mechanism

The CPU throttle mechanism in the XML agent controls the number of tags to process per second. The higher the number of tags that are specified, the higher the CPU utilization and faster response. The lower number of tags means less CPU utilization and slower response.

To configure the number of tags, use this command:

xml agent [tty | ssl] throttle process-rate <*1000-30000*>

Memory Throttle Mechanism

The memory throttle mechanism in the XML agent controls the maximum XML response size in MB. If this size is exceeded, this error message is returned in the XML response.

```
> XML> <?xml version="1.0" encoding="UTF-8"?>
> <Response MajorVersion="1" MinorVersion="0"><Get ErrorCode="0xa367a600"
ErrorMsg="&apos;XML Service Library&apos; detected the &apos;fatal&apos; condition
&apos;The throttle on the memory usage has been reached. Please optimize the request to
query smaller data.&apos;"/></Response>
```

To configure the size of the memory usage per session, use this command:

xml agent [tty | ssl] throttle memory <*100-600*>

The default is 300 MB.

Streaming

As the XML agent retrieves the data from the source, the output of a response is streamed. This process is similar to iterators, but the XML client does not run the GetNext IteratorID to handle large response data size.

Usage Guidelines

Use these guidelines when streaming is used by the client application:

- Iteration must be off.

  xml agent [**tty | ssl**] **iteration off**

- The sub-response block size is a configurable value specific to each transport mechanisms on the router: the XML agent for the dedicated TCP connection and Secure Shell (SSH), Telnet, or Secure Sockets Layer (SSL) dedicated TCP connection.

  Use this command to configure the streaming size. Specify the streaming size in KB. The default is 48 KB.

  xml agent [**tty | ssl**] **streaming on size** <*1-100000*>