



# Cisco XML and Native Data Access Techniques

This chapter describes the various techniques or strategies you can use to structure native data operation requests to access the information needed within the extensible markup language (XML) schema object class hierarchy.

## Available Set of Native Data Access Techniques

The available native data access techniques are as follows:

- Request all data in the configuration hierarchy. See [“XML Request for All Configuration Data” section on page 5-70.](#)
- Request all configuration data for a component. See [“XML Request for All Configuration Data per Component” section on page 5-70.](#)
- Request all data within a container. See [“XML Request for Specific Data Items” section on page 5-72.](#)
- Combine object class hierarchies within a request. See [“XML Request with Combined Object Class Hierarchies” section on page 5-73.](#)
- Use wildcards in order to apply an operation to a set of entries within a table (Match attribute). See [“XML Request Using Wildcarding \(Match Attribute\)” section on page 5-76.](#)
- Repeat naming information in order to apply an operation to multiple instances of an object. See [“XML Request for Specific Object Instances \(Repeated Naming Information\)” section on page 5-78.](#)
- Perform a one-level <Get> in order to “list” the naming information for each entry within a table (Content attribute). See [“XML Request Using Operation Scope \(Content Attribute\)” section on page 5-80.](#)
- Specify the maximum number of table entries to be returned in a response (Count attribute). See [“Limiting the Number of Table Entries Returned \(Count Attribute\)” section on page 5-82.](#)
- Use custom filters to filter table entries (Filter element). See [“Custom Filtering \(Filter Element\)” section on page 5-83.](#)

The actual data returned in a <Get> request depends on the value of the Source attribute as defined in the [“Getting Configuration Data” section on page 2-32.](#)

**Note**

The term “container” is used in this document as a general reference to any grouping of related data, for example, all of the configuration data for a particular Border Gateway Protocol (BGP) neighbor. The term “table” is used more specifically to denote a type of container that holds a list of named homogeneous objects. For example, the BGP neighbor address table contains a list of neighbor addresses, each of which is identified by its IP address. All table entries in the XML API are identified by the unique value of their <Naming> element.

## XML Request for All Configuration Data

Use the empty <Configuration/> tag to retrieve the entire configuration object class hierarchy.

The following example shows how to get the entire configuration hierarchy by specifying the empty <Configuration/> tag.

### Sample XML Client Request to <Get> the Entire Configuration Object Class Hierarchy

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration/>
  </Get>
</Request>
```

### Sample XML Response from the Router

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      .
      .
      .
      response data goes here
      .
      .
      .
    </Configuration>
  </Get>
</Response>
```

## XML Request for All Configuration Data per Component

All the configuration data for a component is retrieved by specifying the highest level tag for the component.

In the following example, all the configuration data for BGP is retrieved by specifying the empty <BGP/> tag.

### Sample XML Client Request for All BGP Configuration Data

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

```

</Get>
</Request>

```

### Sample XML Response from the Router

```

<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        .
        .
        .
        response data goes here
        .
        .
        .
      </BGP>
    </Configuration>
  </Get>
</Response>

```

## XML Request for All Data Within a Container

All data within a containers is retrieved by specifying the configuration or operational object class hierarchy down to the containers of interest, including any naming information as appropriate.

The following example shows how to retrieve the configuration for the BGP neighbor with address 10.0.101.6:

### Sample XML Client Request to Get All Address Family-Independent Configuration Data Within a BGP Neighbor Container

```

<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.6</IPV4Address>
                  </IPAddress>
                </Naming>
              </Neighbor>
            </NeighborTable>
          </BGPEntity>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Request>

```

**Sample XML Response from the Router**

```

<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.6</IPV4Address>
                  </IPAddress>
                </Naming>
                <RemoteAS>6</RemoteAS>
                <EBGPMultihopMaxHopCount>255</EBGPMultihopMaxHopCount>
                <NeighborAFTable>
                  <NeighborAF>
                    <Naming>
                      <AF>IPv4Unicast</AF>
                    </Naming>
                    <Activate>true</Activate>
                    <PrefixListFilterIn>orf</PrefixListFilterIn>
                    <AdvertiseORF>Both</AdvertiseORF>
                  </NeighborAF>
                  <NeighborAF>
                    <Naming>
                      <AF>IPv4Multicast</AF>
                    </Naming>
                    <Activate>true</Activate>
                    <PrefixListFilterIn>orf</PrefixListFilterIn>
                  </NeighborAF>
                </NeighborAFTable>
              </Neighbor>
            </NeighborTable>
          </BGPEntity>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Response>

```

**XML Request for Specific Data Items**

The value of a specific data item (leaf object) can be retrieved by specifying the configuration or operational object class hierarchy down to the item of interest, including any naming information as appropriate.

The following example shows how to retrieve the values of the two data items `<RemoteAS>` and `<EBGPMultihopMaxHopCount>` for the BGP neighbor with address 10.0.101.6:

**Sample XML Client Request for Two Specific Data Items: RemoteAS and EBGPMultihopMaxHopCount**

```

<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>

```

```

<BGP MajorVersion="1" MinorVersion="0">
  <AS>
    <Naming>
      <AS>3</AS>
    </Naming>
    <BGPEntity>
      <NeighborTable>
        <Neighbor>
          <Naming>
            <IPAddress>
              <IPV4Address>10.0.101.6</IPV4Address>
            </IPAddress>
          </Naming>
          <RemoteAS/>
          <EBGPMultihopMaxHopCount/>
        </Neighbor>
      </NeighborTable>
    </BGPEntity>
  </AS>
</BGP>
</Configuration>
</Get>
</Request>

```

### Sample XML Response from the Router

```

<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.6</IPV4Address>
                  </IPAddress>
                </Naming>
                <RemoteAS>6</RemoteAS>
                <EBGPMultihopMaxHopCount>255</EBGPMultihopMaxHopCount>
              </Neighbor>
            </NeighborTable>
          </BGPEntity>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Response>

```

## XML Request with Combined Object Class Hierarchies

Multiple object class hierarchies can be specified in a request. For example, a portion of the hierarchy can be repeated, and multiple instances of a child object class can be included under a parent.

The object class hierarchy may also be compressed into the most “efficient” XML. In other words, it is not necessary to repeat hierarchies within a request.

Before combining multiple operations inside one <Get> tag, the following limitations should be noted for Release 3.0. Any operations that request multiple items of data must be sent in a separate XML request. They include:

- An operation to retrieve all data beneath a container. For more information, see “[XML Request for All Data Within a Container](#)” section on page 5-71.
- An operation to retrieve the list of entries in a table. For more information, see “[XML Request Using Operation Scope \(Content Attribute\)](#)” section on page 5-80.
- An operation which includes a wildcard. For more information, see “[XML Request Using Wildcarding \(Match Attribute\)](#)” section on page 5-76.

If an attempt is made to make such an operation followed by another operation within the same request, the following error will be returned:

```
XML Service Library detected the 'fatal' condition. The XML document which led to this response contained a request for a potentially large amount of data, which could return a set of iterators. The document also contained further requests for data, but these must be sent in a separate XML document, in order to ensure that they are serviced.
```

The error indicates that the operations must be separated out into separate XML requests.

The following two examples illustrate two different object class hierarchies that retrieve the same data: the value of the leaf object <RemoteAS> and <EBGPMultihopMaxHopCount> for the BGP neighbor with the address 10.0.101.6 and all of the configuration data for the BGP neighbor with the address 10.0.101.7:

### Example 1: Verbose Form of a Request Using Duplicated Object Class Hierarchies

#### Sample XML Client Request for Specific Configuration Data Values

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.6</IPV4Address>
                  </IPAddress>
                </Naming>
                <!-- Gets the following two leaf objects for this neighbor -->
                <RemoteAS/>
                <EBGPMultihopMaxHopCount/>
              </Neighbor>
            </NeighborTable>
          </BGPEntity>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Get>
```

```

<Configuration>
  <BGP MajorVersion="1" MinorVersion="0">
    <AS>
      <Naming>
        <AS>3</AS>
      </Naming>
      <BGPEntity>
        <NeighborTable>
          <Neighbor>
            <Naming>
              <IPAddress>
                <!-- Gets all configuration data for this neighbor -->
                <IPV4Address>10.0.101.7</IPV4Address>
              </IPAddress>
            </Naming>
          </Neighbor>
        </NeighborTable>
      </BGPEntity>
    </AS>
  </BGP>
</Configuration>
</Get>
</Request>

```

### Sample XML Response from the Router

```

<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      .
      .
      .
      response data returned here for
      neighbor 10.0.101.6
      .
      .
    </Configuration>
  </Get>
  <Get>
    <Configuration>
      .
      .
      .
      response data returned here
      neighbor 10.0.101.7
      .
      .
    </Configuration>
  </Get>
</Response>

```

## Example 2: Compact Form of a Request Using Compressed Object Class Hierarchies

### Sample XML Client Request

```

<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">

```

```

<AS>
  <Naming>
    <AS>3</AS>
  </Naming>
  <BGPEntity>
    <NeighborTable>
      <Neighbor>
        <Naming>
          <IPAddress>
            <IPV4Address>10.0.101.6</IPV4Address>
          </IPAddress>
        </Naming>
        <!-- Gets the following two leaf objects for this neighbor -->
        <RemoteAS/>
        <EBGPMultihopMaxHopCount/>
      </Neighbor>
      <Neighbor>
        <Naming>
          <IPAddress>
            <!-- Gets all configuration data for this neighbor -->
            <IPV4Address>10.0.101.7</IPV4Address>
          </IPAddress>
        </Naming>
      </Neighbor>
    </NeighborTable>
  </BGPEntity>
</AS>
</Configuration>
</Get>
</Request>

```

### Sample XML Response from the Router

```

<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      .
      .
      .
      response data returned here for both
      neighbors
      .
      .
      .
    </Configuration>
  </Get>
</Response>

```

## XML Request Using Wildcarding (Match Attribute)

Wildcarding of naming information is provided by means of the Match attribute. Match="\*" can be used on any Naming attribute within a <Get> or <Delete> operation to effectively specify a wildcarded value for that attribute. The operation applies to all instances of the requested objects.

"\*" is the only value supported for Match, though other wildcarding or matching specifications may be supported in the future. The Match attribute is comprehensively supported for table entries in the <Configuration> namespace, but in the <Operational> space the limitation currently exists that nonwildcarded naming information cannot appear in the hierarchy below wildcarded naming information.

**Note**

Although partial wildcarding of NodeIDs is not available in XML, each element of the NodeID has to be wildcarded, similar to the support on the CLI of \*/\*/ as the only wildcards supported for locations.

The following example shows how to use the Match attribute to get the <RemoteAS> value for all configured BGP neighbors.

**Sample XML Client Request Using the Match Attribute Wildcarding**

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable>
              <Neighbor>
                <Naming>
                  <IPAddress Match="*" />
                </Naming>
                <RemoteAS/>
              </Neighbor>
            </NeighborTable>
          </BGPEntity>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Request>
```

**Sample XML Response from the Router**

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.1</IPV4Address>
                  </IPAddress>
                </Naming>
                <RemoteAS>1</RemoteAS>
              </Neighbor>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.2</IPV4Address>
                  </IPAddress>
                </Naming>
                <RemoteAS>2</RemoteAS>
              </Neighbor>
            </NeighborTable>
          </BGPEntity>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Response>
```

```

    </Neighbor>
  <Neighbor>
    <Naming>
      <IPAddress>
        <IPv4Address>10.0.101.3</IPv4Address>
      </IPAddress>
    </Naming>
    <RemoteAS>3</RemoteAS>
  </Neighbor>
  .
  .
  .
  data for more neighbors
  returned here
  .
  .
  .
</NeighborTable>
</BGPEntity>
</AS>
</BGP>
</Configuration>
</Get>
</Response>

```

## XML Request for Specific Object Instances (Repeated Naming Information)

Wildcarding allows the client application to effectively specify all instances of a particular object. Similarly, the client application might have a need to specify only a limited set of instances of an object. Specifying object instances can be done by simply repeating the naming information in the request.

The following example shows how to retrieve the address independent configuration for three different BGP neighbors, that is, the neighbors with addresses 10.0.101.1, 10.0.101.6, and 10.0.101.8, by repeating the naming information, once for each desired instance.

### Sample XML Client Request Using Repeated Naming Information for BGP <NeighborAddress> Instances

```

<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPv4Address>10.0.101.1</IPv4Address>
                  </IPAddress>
                </Naming>
              </Neighbor>
            </NeighborTable>
            <NeighborTable>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPv4Address>10.0.101.6</IPv4Address>

```

```

        </IPAddress>
      </Naming>
    </Neighbor>
  </NeighborTable>
  <NeighborTable>
    <Neighbor>
      <Naming>
        <IPAddress>
          <IPv4Address>10.0.101.8</IPv4Address>
        </IPAddress>
      </Naming>
    </Neighbor>
  </NeighborTable>
</BGPEntity>
</AS>
</BGP>
</Configuration>
</Get>
</Request>

```

### Sample XML Response from the Router

```

<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPv4Address>10.0.101.1</IPv4Address>
                  </IPAddress>
                </Naming>
                .
                .
                .
                data returned for 1st neighbor
                .
                .
                .
              </Neighbor>
            <Neighbor>
              <Naming>
                <IPAddress>
                  <IPv4Address>10.0.101.6</IPv4Address>
                </IPAddress>
              </Naming>
              .
              .
              .
              data returned for 2nd neighbor
              .
              .
              .
            </Neighbor>
          <Neighbor>
            <Naming>
              <IPAddress>

```

```

        <IPV4Address>10.0.101.6</IPV4Address>
      </IPAddress>
    </Naming>
    .
    .
    .
    data returned for 3rd neighbor
    .
    .
  </Neighbor>
</NeighborTable>
</BGPEntity>
</AS>
</BGP>
</Configuration>
</Get>
</Response>

```

## XML Request Using Operation Scope (Content Attribute)

The Content attribute is used on any table element in order to specify the scope of a <Get> operation. [Table 5-1](#) describes the content attribute values are supported.

**Table 5-1 Content Attributes**

| Content Attribute | Description  |
|-------------------|--|
| All               | Use to get all leaf items and their values. All is the default when the Content attribute is not specified on a table element.       |
| Entries           | Use to get the Naming information for each entry within a specified table object class. Entries provides a one-level get capability. |

If the Content attribute is specified on a nontable element, it is ignored. Note also that the Content and Count attributes can be used together on the same table element.

The following example displays the Content attribute that is used to list all configured BGP neighbors:

### Sample XML Client Request Using the All Content Attribute

```

<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable Content="Entries"/>
          </BGPEntity>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Request>

```

**Sample XML Response from the Router**

```

<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable Content="Entries">
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.1</IPV4Address>
                  </IPAddress>
                </Naming>
              </Neighbor>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.2</IPV4Address>
                  </IPAddress>
                </Naming>
              </Neighbor>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.3</IPV4Address>
                  </IPAddress>
                </Naming>
              </Neighbor>
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.4</IPV4Address>
                  </IPAddress>
                </Naming>
              </Neighbor>
              .
              .
              .
              more neighbors returned here
              .
              .
            </NeighborTable>
          </BGPEntity>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Request>

```

## Limiting the Number of Table Entries Returned (Count Attribute)

The Count attribute is used on any table element within a <Get> operation to specify the maximum number of table entries to be returned in a response. When the Count attribute is specified, the naming information within the request is used to identify the starting point within the table, that is, the first table entry of interest. If no naming information is specified, the response starts at the beginning of the table.

For a table whose entries are containers, the Count attribute can be used only if the Content attribute is also specified with a value of Entries. This restriction does not apply to a table whose children are leaf nodes.

As an alternative to the use of the Count attribute, the XML interface supports the retrieval of large XML responses in blocks through iterators. For more information on iterators, see [Chapter 7, “Cisco XML and Large Data Retrieval \(Iterators\).”](#)

The following example shows how to use the Count attribute to retrieve the configuration information for the first five BGP neighbors starting with the address 10.0.101.1:

### Sample XML Client Request Using the Count Attribute

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable Count="5">
              <Neighbor>
                <Naming>
                  <IPAddress>
                    <IPV4Address>10.0.101.1</IPV4Address>
                  </IPAddress>
                </Naming>
              </Neighbor>
            </NeighborTable>
          </BGPEntity>
        </AS>
      </BGP>
    </Configuration>
  </Get>
</Request>
```

### Sample XML Response from the Router

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Configuration>
      <BGP MajorVersion="1" MinorVersion="0">
        <AS>
          <Naming>
            <AS>3</AS>
          </Naming>
          <BGPEntity>
            <NeighborTable Count="5">
              <Neighbor>
                <Naming>
                  <IPAddress>
```

```

        <IPV4Address>10.0.101.1</IPV4Address>
    </IPAddress>
</Naming>
.
.
.
    data for 1st neighbor returned
    here
.
.
.
</Neighbor>
<Neighbor>
    <Naming>
        <IPAddress>
            <IPV4Address>10.0.101.2</IPV4Address>
        </IPAddress>
    </Naming>
.
.
.
    data returned for 2nd neighbor
    here
.
.
.
</Neighbor>
.
.
.
    data returned for remaining
    neighbors here
.
.
.
</NeighborTable>
</BGPEntity>
</AS>
</BGP>
</Configuration>
</Get>
</Response>

```

## Custom Filtering (Filter Element)

Some of the tables from the operational namespace support the selection of rows of interest based on predefined filtering criteria. Filters can be applied to such tables in order to reduce the number of table entries retrieved in a request.

Client applications specify filtering criteria for such tables by using the <Filter> tag and including the filter specific parameters as defined in the XML schema definition for that table. If no table entries match the specified filter criteria, the response contains the object class hierarchy down to the specified table, but does not include any table entries. The Content attribute can be used with a filter to specify the scope of a <Get> request.

In the following example, the filter <BGP\_ASFilter> is used to retrieve operational information for all neighbors in autonomous system 6:

### Sample XML Client Request Using Filtering

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Get>
    <Operational>
      <BGP MajorVersion="1" MinorVersion="0">
        <NeighborTable>
          <Filter>
            <BGP_ASFilter>
              <AS>6</AS>
            </BGP_ASFilter>
          </Filter>
        </NeighborTable>
      </BGP>
    </Operational>
  </Get>
</Request>
```

### Sample Filtered XML Response from the Router

```
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
  <Get>
    <Operational>
      <BGP MajorVersion="1" MinorVersion="0">
        <NeighborTable>
          <Filter>
            <BGP_ASFilter>
              <AS>6</AS>
            </BGP_ASFilter>
          </Filter>
          <Neighbor>
            .
            .
            .
            data for 1st neighbor returned here
            .
            .
            .
          </Neighbor>
          <Neighbor>
            .
            .
            .
            data for 2nd neighbor returned here
            returned here
            .
            .
            .
          </Neighbor>
            .
            .
            .
            data for remaining neighbors returned
            here
            .
            .
            .
          </NeighborTable>
        </BGP>
```

```
    </Operational>  
  </Get>  
</Response>
```

