Troubleshoot XML for Cisco IOS XR

Contents

Introduction

This document describes XML memory throttle issues seen in service requests and Cisco tools.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Cisco IOS® XR platform
- XML (Extensible Markup Language) infrastructure
- Common Object Request Broker Architecture (CORBA)
- Element Management System (EMS)
- External Data Manager (EDM)

Components Used

The information in this document is based on ASR9000.

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Throttle Memory Issues

A case was opened with these questions:

1. What is the max memory configurable on the router?
2. Is there a way to break large XML replies?

Answer:

1. The max memory change depending on the version that the RSP/node is running (versions for cXR and eXR)

In 64-bit version (eXR). You have one throttle memory size:

```
RP/0/RSP1/CPU0:XR#show version
Wed Jul 26 21:10:16.761 IST
Cisco IOS XR Software, Version 7.1.3
Copyright (c) 2013-2020 by Cisco Systems, Inc.

Build Information:
 Built By     : gopalk2
 Built On     : Thu Nov 26 10:51:48 PST 2020
 Built Host   : iox-ucs-027
 Workspace    : /auto/srcarchive17/prod/7.1.3/asr9k-x64/ws
 Version      : 7.1.3
 Location     : /opt/cisco/XR/packages/
 Label        : 7.1.3

cisco ASR9K () processor
System uptime is 2 weeks 17 hours 22 minutes


RP/0/RSP1/CPU0:XR# configuration
RP/0/RSP1/CPU0:XR(config)#xml agent throttle ?
  memory        Memory usage
  process-rate  Process rate
RP/0/RSP1/CPU0:XR(config)#xml agent throttle memory ?
  <100-1024>  Size of the memory usage in Mbytes per session (default 300 Mbytes)
```

In 32-bit version (cXR). You have another limit:

```
RP/0/RSP0/CPU0:XR#show version
Cisco IOS XR Software, Version 6.4.2[Default]
Copyright (c) 2020 by Cisco Systems, Inc.

ROM: System Bootstrap, Version 10.59(c) 1994-2014 by Cisco Systems,  Inc.
<snip>

Configuration register on node 0/RSP0/CPU0 is 0x102
Boot device on node 0/RSP0/CPU0 is disk0:
```

```
RP/0/RSP0/CPU0:XR# configuration
RP/0/RSP0/CPU0:XR(config)#xml agent throttle memory ?
  <100-600>  Size of the memory usage in Mbytes per session (default 300 Mbytes)
```

Note: The default in either version is 300 Mbytes

2. There are different ways to break the XML reply:

- Iterators:

  When Iterators are configured you segment of the XML total response in specific windows size. The window have the Iterator size. For example if the response is 1 Gb, and and Iterator size of 500 Mb, you segment the XML into 2 replies.

  This method changes the response by adding (based on the Iterator size) GetNext operations (what the response needs).

  There can be up to 10 Iterators for a session.

- Throttling (process-rate):

  This feature limits the memory consumed by the XML process. In case a process overpass the memory, it replies with the error message: "The throttle on the memory usage has been reached".

- Streaming:

  Similar to Iterators, Streaming segments the XML response into specific windows. The difference is the response, it removes the GetNext operation and the Iterator ID. The XML sends the streamed packages and the client build the response when the stream ends.

Further Problem Description

For use cases where automation is needed, use the tool pyIOSXR. This automation tool is an XML agent, it helps to issue some show commands and, overall, connects to the device.

Every time you send a big request with this agent, an error is displayed:

Note: Only registered Cisco users can access internal Cisco tools and information.

```
RESPONSE ERROR: 0xa367a600 'XML Service Library' detected the 'fatal' condition 'The throttle on the mem
```

With error message displaying "optimize the request to query smaller data", you can configure the agent XML API with Iterators. This, in theory, allows segmenting the response.

When Iterators were configured, the same error message displayed: "RESPONSE ERROR...Please optimize the request to query smaller data".

When an error is displayed, the next step is to understanding why the feature, in this case Iterators, did not work for the query.

The pyIOSXR recommends the XML agent to be properly enabled in the device, meaning that the API does not allows the usage of Iterators.

The next step is to test the second option: Streaming.

Note: pyIOSXR does not allow us to work with different headers other than the basic XML query which have the next elements:

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

Streaming and Iterators add a header in the queries. Streaming adds a stream ID that help the system to build the full response. Iterators add the GET_Next and Iterator ID.

Testing streaming also showed the same error message as Iterators.

Solution

In the section, More Recommendation to do Queries to the Device, are more tools to sort these issues out. Wildcard is one of them. Wildcard is the solution for the memory throttle limit.

Wildcard builds an specific query to avoid requesting unnecessary information. For example, for BGP information use the **show route bgp** command instead of the generic **show route** command. This example applies to the XML queries and logic. Requesting bulk information to the system can generate memory and processing issues.

When the case was opened this query was used:

```
<?xml version="1.0"  encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
    <Get>
        <Operational>
            <BGP  MajorVersion="36" MinorVersion="1">
               <InstanceTable>
                    <Instance>
                        <Naming>
                            <InstanceName>
                                DEFAULT
                            </InstanceName>
                        </Naming>
                        <InstanceActive>
                            <VRFTable>
                                <VRF>
                                    <Naming>
                                        <VRFName>
                                            VRF_NAME
                                        </VRFName>
                                    </Naming>
                                    <AFTable>
                                        <AF>
                                            <Naming>
                                                <AFName>
                                                    NAME
                                                </AFName>
                                            </Naming>
                                            <NetworkTable>
                                            </NetworkTable>
                                        </AF>
                                    </AFTable>
                                </VRF>
                            </VRFTable>
                        </InstanceActive>
                    </Instance>
               </InstanceTable>
            </BGP>>
        </Operational>
    </Get>
</Request>
```

This query targets the full BGP tables. For this non-specific request, the response was almost 2.2 Gb, therefore, the memory throttle limit is reached.

To fix it, an specific query is required, this allows the system to process the query, and return the information.

XML in Cisco IOS XR

Note: In case further information is required, specifications or other commands, please click the next link XML Programming guide: [XR XML programming guide]

The XML defines how data is displayed and structured. This is a way to parse what the computer understands as bits, and displays structured, standardized information.

XML has this structure:

```
<init>
    <body>
        <message>This is an example</message>
    </body>
</init>
```

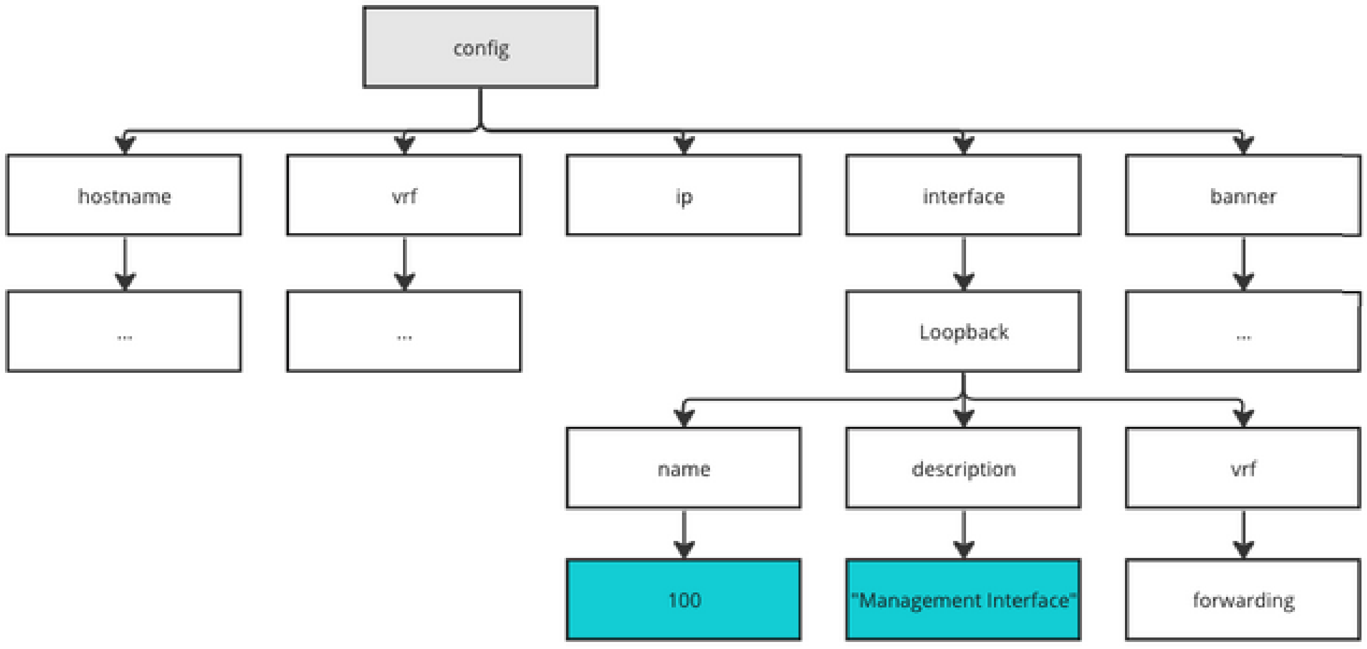
All the tags have two parts, the opening tag (<init>) and the closing tag (</init>). If this structure is not used, XML cannot understand where a tag ends.

XML is built of two entities:

- Containers
- Leafs

An analogy for this structured data is comparing XML data as a tree. A container is a branch and every branch has stubbed leafs. Leafs do not contain any other than information.

For example, the next tree has the root in grey, containers in white, and leafs in blue.



Using XML API

To test/utilize the XML API the first thing you need is a query.

1. Query has a header:

```
<?xml version="1.0" encoding="UTF-8”?>
```

Note: This is the only tag that do not need a closing tag.

2. You need to add a tag for the request. In the request, you need to specify the version.

```
<Request MajorVersion="1" MinorVersion="0">
```

3. With the header, request, and version, the body continues with any XML operation that the API has:

```
<Operation>
<Operation_1>
<Operation_2>
…
<Operation_n>
```

Note: Notice that requests include different operations in each query. It is not necessary to perform a request per operation.

The next example displays a request with all the required information:

```
<?xml version="1.0" encoding="UTF-8”?>
<Request MajorVersion="1" MinorVersion="0">
    <Operation></Operation>
    <Operation_1></Operation_1>
    <Operation_2></Operation_2>
    …
    <Operation_n></Operation_n>
</Request>
```

Operations Supported by XML API

Cisco IOS XR supports 5 operations that allow users to interact with the information the XML schema has:

1. Native Data Operations:

- <Get></Get>: To collect action data items (Data that leaf has).
- <Set></Set>: To configure, change, or add data items.
- <Delete></Delete>: Eliminate one or more data items.
- <GetVersionInfo></GetVersionInfo>: Operation that retrieves the major and minor version numbers of the requested components.
- <GetDataSpaceInfo></GetDataSpaceInfo>: This operational tags display the leaf names that are mapped to the containers.

2. CLI operations:

- <CLI><CLI/>: Tag that allows us to issue a configuration request as a CLI command.
- <GetNext><GetNext/>:If the data is bigger than the block size, this tag is added to the next segmented reply.

Offered XML Schema Services

XML API implements the next namespaces:

Note: In case you need to see root objects, visit: Cisco IOS-XR XML Network Management.

Caution: Be advised that each namespace oversees different data and operations.

- Configuration: Operation that allows the Get, Set and Delete actions.
- Operational: Like CLI show commands.
- Action: Support the Set action. With this operation you can access action data, like clear commands and other similar commands.
- AdminOperational: Access to Admin operational data.
- AdminAction: Access to the administration action data.
- AdminConfiguration: Access to the administration configuration data.

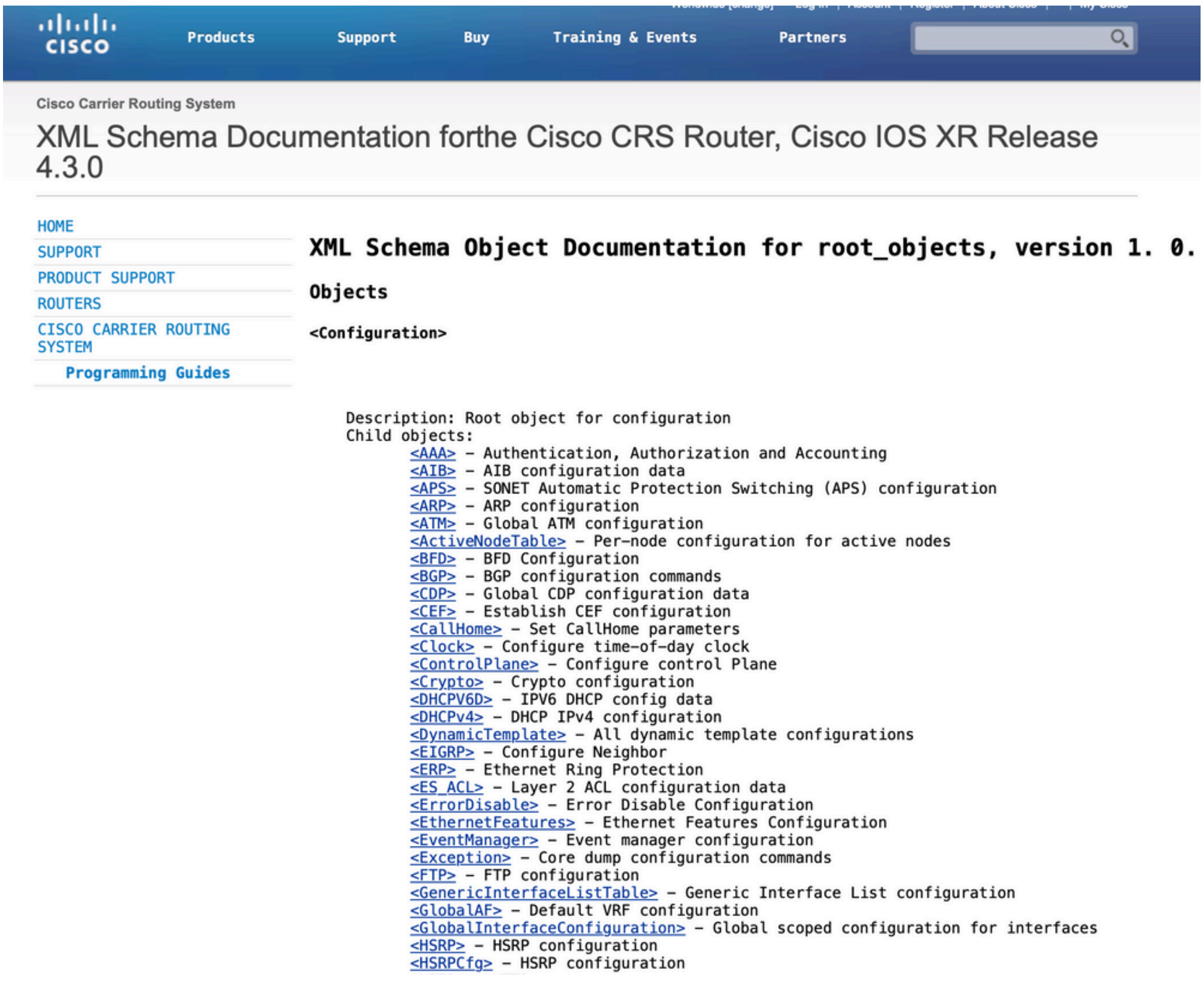
Building a Query

There are two different paths to travel. Once you selected the namespaces for the query you need to use a schema to work with:

1. XML Schema Documentation

These options provides a full XML tree that shows containers and leafs. For documentation click CRS XML Schemas.

The documentation page displays.



Cisco Carrier Routing System

XML Schema Documentation forthe Cisco CRS Router, Cisco IOS XR Release 4.3.0

XML Schema Object Documentation for root_objects, version 1. 0.

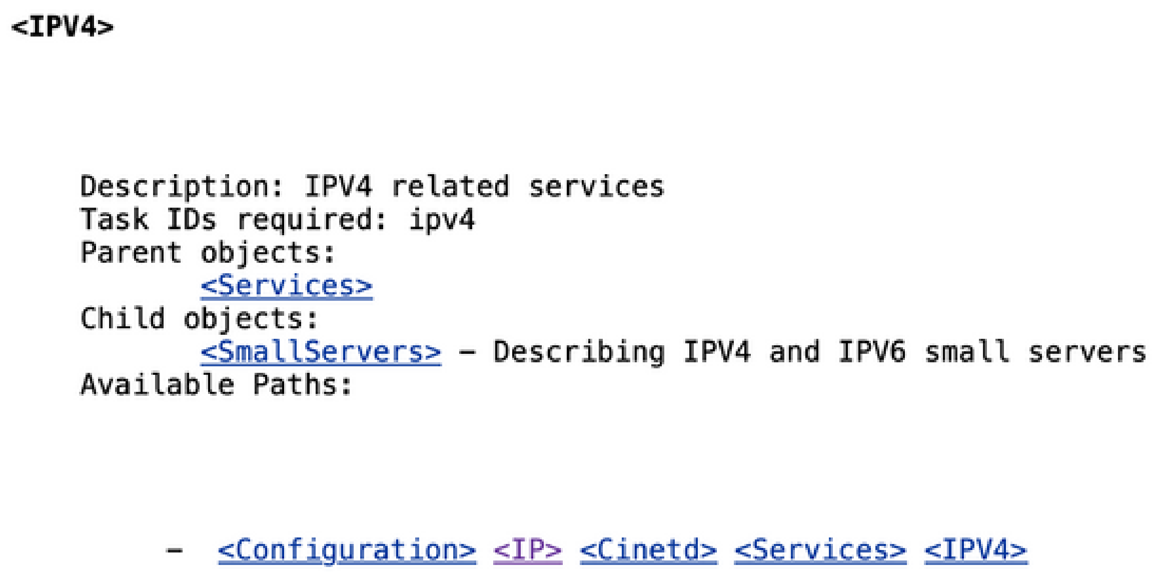
Objects

<Configuration>

```
    Description: Root object for configuration
    Child objects:
            <AAA> - Authentication, Authorization and Accounting
            <AIB> - AIB configuration data
            <APS> - SONET Automatic Protection Switching (APS) configuration
            <ARP> - ARP configuration
            <ATM> - Global ATM configuration
            <ActiveNodeTable> - Per-node configuration for active nodes
            <BFD> - BFD Configuration
            <BGP> - BGP configuration commands
            <CDP> - Global CDP configuration data
            <CEF> - Establish CEF configuration
            <CallHome> - Set CallHome parameters
            <Clock> - Configure time-of-day clock
            <ControlPlane> - Configure control Plane
            <Crypto> - Crypto configuration
            <DHCPV6D> - IPV6 DHCP config data
            <DHCPv4> - DHCP IPv4 configuration
            <DynamicTemplate> - All dynamic template configurations
            <EIGRP> - Configure Neighbor
            <ERP> - Ethernet Ring Protection
            <ES_ACL> - Layer 2 ACL configuration data
            <ErrorDisable> - Error Disable Configuration
            <EthernetFeatures> - Ethernet Features Configuration
            <EventManager> - Event manager configuration
            <Exception> - Core dump configuration commands
            <FTP> - FTP configuration
            <GenericInterfaceListTable> - Generic Interface List configuration
            <GlobalAF> - Default VRF configuration
            <GlobalInterfaceConfiguration> - Global scoped configuration for interfaces
            <HSRP> - HSRP configuration
            <HSRPCfg> - HSRP configuration
```

This page displays objects, containers, and children. Each object contains a child object. If the child object does not contain any other container, it is considered as a leaf.

You can click on the child, and the information to build a query is displayed. For example for IPv4:



```
<IPV4>

    Description: IPV4 related services
    Task IDs required: ipv4
    Parent objects:
            <Services>
    Child objects:
            <SmallServers> - Describing IPV4 and IPV6 small servers
    Available Paths:

         -  <Configuration> <IP> <Cinetd> <Services> <IPV4>
```

In the previous image, for any IPv4 query the **Configuration** operation can be run, targeting the container **IP**, in the container **Cinetd**, in the container **Services**, and then, finally, in the leaf **IPv4**.

Note: Any operation that needs to be a leaf in order to perform the query.

The query would be the next:

```
<?xml version="1.0"  encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
    <Get>
        <Configurations>
            <IP>
                <Cinetd>
                    <Services>
                        <IPv4/>
                    </Services>
                </Cinetd>
            </IP>
        </Configurations>
    </Get>
</Request>
```

2. XML API

The XML API is embedded in Cisco IOS XRl. To see the XML tree use the **show xml schema** command. By issuing the command, the whole XML schema is seen in a CLI similar to Linux.

In this CLI there are the next actions:

?:Just like **--help**

LS**:** List the containers/leafs in the branch user is.

<operation>: By issuing **?** you can check for the operations and commands can be issued:

Some operations are:

- config

- adminconfig

- cd

- list

- oper

- action

CD: Change Directory/container.

Example shows the next:

```
RP/0/RP0/CPU0:XR#show xml schema
Username:admin
```

```
Password:

xml-schema[config]:> ?

config                   oper                     action
adminconfig              adminoper                adminaction
cd                       pwd                      classinfo
list                     ls                       datalist
walk                     walkdata                 get
hierarchy                quit                     exit
help
xml-schema[config]:> ls
[container]                   RIP
[container]                   TCL
[container]                   LawfulIntercept
[container]                   ErrorDisable
[container]                   PerfMgmt
[container]                   RCC
[container]                   FrequencySynchronization
[container]                   HwModuleProfileConfig
[container]                   MPLSStatic
[container]                   XML
[container]                   Tpa
[container]                   MLD
[leaf]                        RPIsolationEnabled
[leaf]                        RPIsolationMultiple
[container]                   AMT
[container]                   PriorityFlowControlWatchdog
[container]                   SSH
[container]                   BNG_PBR
<snip>
```

Warning: Note that credentials are needed to log into to the device. These credentials are local to the device and require root-ls/admin profiles.

The next example shows how to build a query using the XML API. For example, the query has to check if the XML agent is enable:

```
xml-schema[config]:> ls
[container]                 RIP
[container]                 TCL
[container]                 LawfulIntercept
[container]                 ErrorDisable
[container]                 PerfMgmt
[container]                 RCC
[container]                 FrequencySynchronization
[container]                 HwModuleProfileConfig
[container]                 MPLSStatic
[container]                 XML   >>> Here
[container]                 Tpa
[container]                 MLD
[leaf]                      RPIsolationEnabled
```

```
[leaf]                        RPIsolationMultiple
[container]                   AMT
[container]                   PriorityFlowControlWatchdog
[container]                   SSH
[container]                   BNG_PBR
[container]                   L2TP
[container]                   Exception
[container]                   IP_RAW
[container]                   MSTAG
[container]                   FpdXRConfig

xml-schema[config]:> cd XML

xml-schema[config]:XML> ls
[container]                   Agent

xml-schema[config]:XML> cd Agent

xml-schema[config]:XML.Agent> ls
[container]                   Default
[container]                   SSL
[container]                   TTY

xml-schema[config]:XML.Agent> cd TTY

xml-schema[config]:XML.Agent.TTY> ls
[leaf]                        Enable   >>> Leaf of interest.
[leaf]                        IterationSize
[leaf]                        StreamingSize
[container]                   Throttle
[container]                   Session
```

Notice the next prompt:

```
xml-schema[config]:XML.Agent.TTY>
```

The previous prompt displays the XML tree. With this information the query is:

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

Testing XML Queries

Once the query is built, the next step is to test it. To test it, you can issue the **xml echo format** command in the CLI. This action can be archived in the same device.

Tip: Be advise that after entering the previous command, users can copy/paste the query and hit enter. The device displays the response to the query.

```
RP/0/RP0/CPU0:XR#xml echo format
XML>
XML> <?xml version="1.0"  encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
    <Get>
        <Configuration>
            <XML>
                <Agent>
                    <TTY></TTY>
                </Agent>
            </XML>
        </Configuration>
    </Get>
</Request>    >>> Hit enter here. Immediately getting the reply.

! Reply:
<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
    <Get>
        <Configuration>
            <XML MajorVersion="1" MinorVersion="4">
                <Agent>
                    <TTY>
                        <Enable>
                            true  >>> This is the requested value.
                        </Enable>
                    </TTY>
                </Agent>
            </XML>
        </Configuration>
    </Get>
    <ResultSummary ErrorCount="0"/>
</Response>
XML> exit
```

The router displays response and ResultSummary.

Note: This XML response does not display nor require any throttle memory to be configured. The response, as it is not using any SSH/TTY, is requested no matter the size. One thing is the XML client and other the embedded XML echo command, the behavior can change.

XML Deep Dive in Cisco IOS XR

Cisco IOS XR is a modular system that requires different packages to work. XML is consider to be in the Management Plane Subsytems.

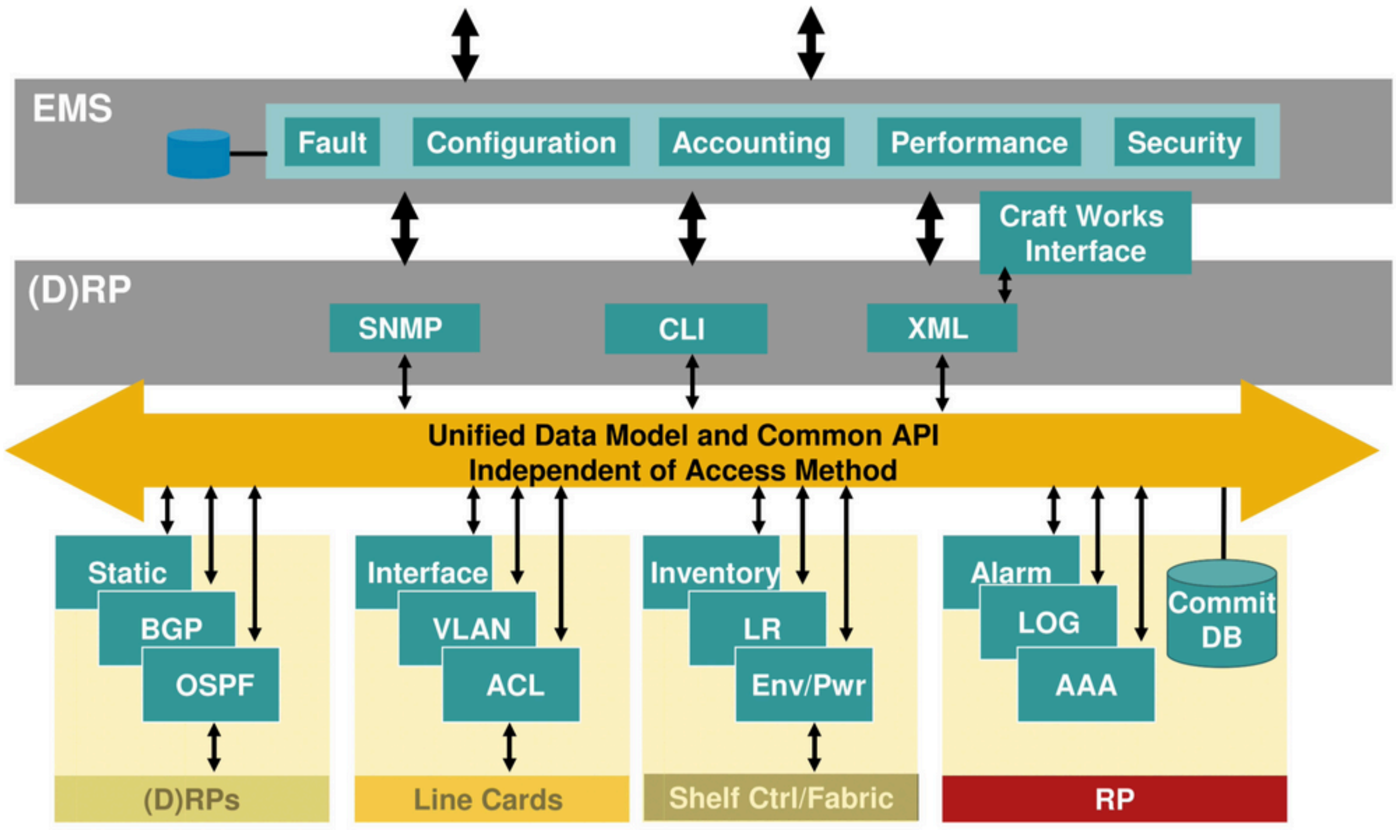
When XR initiated there were 2 different ways to utilize XML in the device:

1. CORBA Agent (using SSL).
2. SSH Agent.

Note: CORBA has been deprecated after version 3.7.

CORBA as a legacy protocol used SSL which is found in the Security Packages. In the other hand, SSH method needs the Manageability Package to be used.

The infra is distributed in this way:



From the previous image, the main process of XML is seen in the Router and Switch Processor (RSP). These processes have a common API that oversees all the information from the other processes in the device, which are:

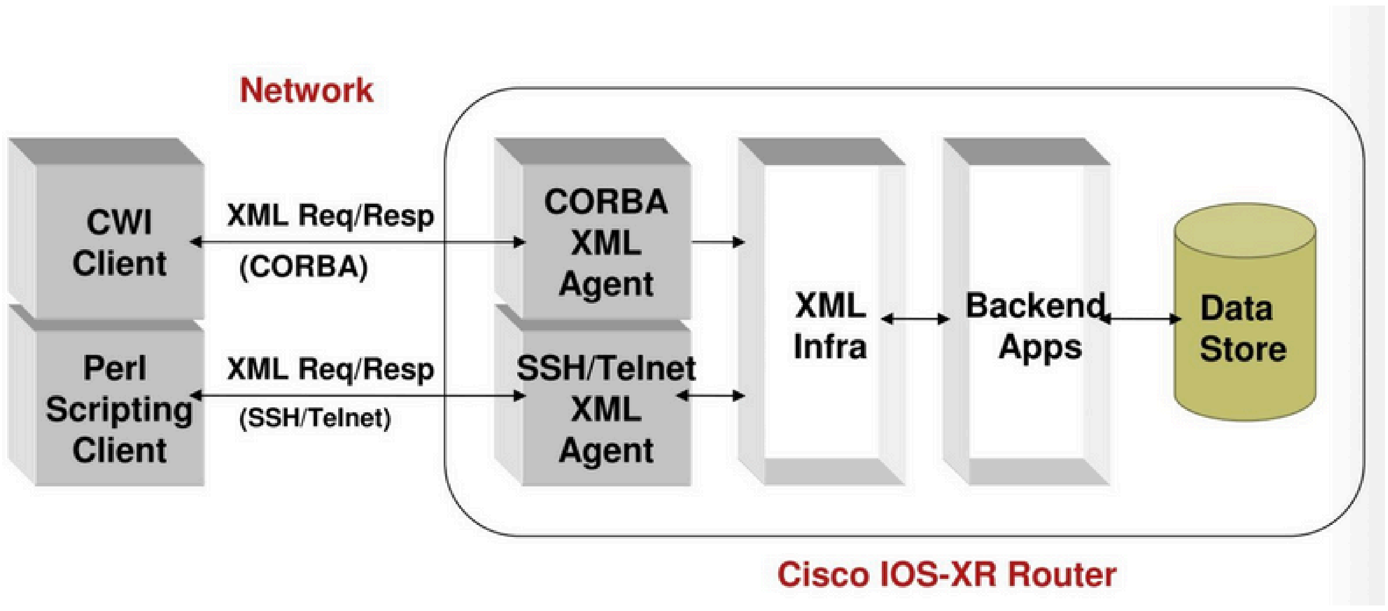
- From the Linecard:
  - Stats Collector
  - Stats Server
- From the Router Processor:
  - Stats Manager.
  - SysDB.
  - TCP (for XML API is used TCP 38751)
  - NetIO

Note: To see more information about this, check the XML Errors in Cisco IOS XR section.

Depending on the request, one or more processes are triggered. When developing automation tools, if the response is bigger than the block size, the response cannot succeed. If a process takes a long time to reply, it can generate EDM logs, shutdown, or affect services.

API Interaction with Client

The next image shows the interaction:



To configure/enable XML agent on the device, use:

```
RP/0/RP0/CPU0:XR#config
RP/0/RP0/CPU0:XR(config)#xml agent tty
RP/0/RP0/CPU0:XR(config-xml-tty)#commit
```

The system needs to agree with the client in these areas:

1. Versions
2. Schema

Versions can be defined in 2 areas:

- Request tag:

```
<Request Major Version="1" MinorVersion="0">
```

- Major component tag:

Which applies to the specific component:

```
<BGP MajorVersion="1" MinorVersion="0">
```

Note: Remember that you can check the schema version with the GetVersionInfo operation applied to the container.

- Minor version update: Any addition to the XML schema, like adding a new data item.
- Major version update: Semantic changes, deletions to the schema or component, and so on.

You can check the version. The next example shows how to do it:

Note: The XML operations are GET, SET, and so on.

```
<?xml version="1.0" encoding="UTF-8"?>
<Request Major Version="1" MinorVersion="0">
<GetVersionInfo>
<Operation/>
</GetVersionInfo>
</Request>
```

Use the **xml echo formal** command and then add the tag GetVersionInfo to the query to find the version being run on the router. as shown here:

```
RP/0/RSP0/CPU0:XR#xml echo format
Mon Jul 31 13:53:50.993 UTC
XML> <?xml version="1.0"  encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
    <GetVersionInfo>
```

```
            <Configuration>
                <XML>
                    <Agent>
                    </Agent>
                </XML>
            </Configuration>
    </GetVersionInfo>
</Request> >>> This is the request

<?xml version="1.0" encoding="UTF-8"?>
<Response MajorVersion="1" MinorVersion="0">
    <GetVersionInfo>
        <Configuration>
            <XML MajorVersion="1" MinorVersion="4">
                <Agent>
                    <Default>
                        <VRFTable>
                            <VRF>
                                <ApplyGroup MajorVersion="2" MinorVersion="3"/>
                                <ExcludeGroup MajorVersion="2" MinorVersion="3"/>
                                <ApplyGroupAppend MajorVersion="2" MinorVersion="3"/>
                                <ApplyGroupRemove MajorVersion="2" MinorVersion="3"/>
                            </VRF>
                        </VRFTable>
                    </Default>
                    <SSL>
                        <VRFTable>
                            <VRF>
                                <ApplyGroup MajorVersion="2" MinorVersion="3"/>
                                <ExcludeGroup MajorVersion="2" MinorVersion="3"/>
                                <ApplyGroupAppend MajorVersion="2" MinorVersion="3"/>
                                <ApplyGroupRemove MajorVersion="2" MinorVersion="3"/>
                            </VRF>
                        </VRFTable>
                    </SSL>
                </Agent>
            </XML>
        </Configuration>
    </GetVersionInfo>
    <ResultSummary ErrorCount="0"/>
</Response>
XML>
```

Note: Request displays all the running versions in the major component and also displays the version in the containers within it.

XML Errors in Cisco IOS XR

If the path is correct, each XML API request, displays the requested information.

When the Query is Wrong

The router displays three different messages:

- **ItemNotFound**

  This message is displayed each time that a GET operation has an empty response.

- **ItemNotFoundBelow**

  GET operation does not contain this operation in the XML Schema.

- **NotFound**

  The requested element cannot be find the the element level.

Error Types

1.Transport: Errors in this category include anything between the XML agent/client communication. This means that any SSH interaction or issues can happen in the transport. Therefore, to check these sort of issues, it is suggested to check the SSH traces to check for any issue with authentication, port, and so on.

2. XML Parser: Any issue with the format and syntax, issues in the response sent or in the query. These issues usually send the reason for failure when an error occurs.

For example:

```
ERROR: 0xa367a600 'XML Service Library' detected the 'fatal' condition 'The throttle on the memory usag
```

3. XML Schema: Any schema different than the router schema. To address these issues, check the CLI schema and versions.

For example:

```
ErrorCode="0x43688400" ErrorMsg="&apos;XMLMDA&apos; detected the &apos;warning&apos; condition &apos;An
```

4. Operation Processing: When you configure the device these issues can occur. To resolve this issues you need to troubleshoot the processes, for example commit, sysdb, and so on.

Note: Error information is added in the operation element level. This is coded in the form of ErrorCode (32-bit int) and Errormsg attributes.

More Recommendations

Other useful techniques:

1. Wildcarding: This is the also known as Specific Queries.
2. Batching: Combining several techniques or operations in a single request (Best-effort operations).
3. Custom filtering: If the schema allows it, to help selection of rows in tables.