



Cisco XML API Overview

This chapter contains the following sections:

- [Introduction, page 1-19](#)
- [Cisco Management XML Interface, page 1-20](#)
- [Cisco XML API and Router System Features, page 1-21](#)
- [Cisco XML API Tags, page 1-22](#)

Introduction

The *Cisco IOS XR XML API Guide* explains how to use the Cisco XML API to configure routers or request information about configuration, management, or operation of the routers. The goal of this guide is to help router customers write client applications to interact with the Cisco XML infrastructure on the router, and to also use the Management XML API to build custom end-user interfaces for configuration and information retrieval and display.

The extensible markup language (XML) application programming interface (API) provided by the router is an interface used for rapid development of client applications and perl scripts to manage and monitor the router. The XML interface is specified by the XML schemas. The XML API provides a mechanism for router configuration and monitoring utilizing an exchange of XML formatted request and response streams.

Client applications can be used to configure the router, or to request status information from the router, by encoding a request in XML API tags and sending it to the router. The router processes the request and sends the response to the client by again encoding the response in XML API tags. This guide describes the XML requests that can be sent by external client applications to access router management data, and also details the responses to the client by the router.

The XML API readily supports available transport layers including the Common Object Request Broker Architecture (CORBA) and the terminal-based protocols Telnet and Secure Shell (SSH).

Customers use a variety of vendor-specific command line interface (CLI) scripts to manage their routers because no alternative programmatic mechanism is available. In addition, a common framework has not been available to develop CLI scripts. In response to this need, the XML API provides the necessary common framework for rapid development, deployment, and maintenance of router management.



Note

The XML API code is available for use on any Cisco platform that runs Cisco IOS XR software.

Definition of Terms

Table 1-1 defines the words, acronyms, and actions used throughout this guide.

Table 1-1 Definition of Terms

Term	Description
AAA	Authentication, authorization, and accounting
CLI	Command-line interface
CORBA	Common Object Request Broker Architecture
CWI	Craft Works Interface
SSH	Secure Shell
XML	Extensible markup language
XML agent	A process on the router that receives XML requests by XML clients, and is responsible to carry out the actions contained in the request and to return an XML response to the client.
XML client	An external application that sends XML requests to the router and receives XML responses to those requests.
XML operation	A portion of an XML request that specifies an operation that the XML client wants the XML agent to perform.
XML operation provider	The code that carries out a particular XML operation including parsing the operation XML, performing the operation, and assembling the operation XML response.
XML request	An XML document sent to the router containing a number of requested operations to be carried out.
XML response	The response to an XML request.
XML schema	An XML document specifying the structure and possible contents of XML elements that can be contained in an XML document.

Cisco Management XML Interface

The following information is listed regarding the Cisco Management XML interface:

- High-level structure of the XML request and response streams
- Operation tag types and usage, including their XML format and content

- How to use XML to configure the router:
 - Using the two-stage “target configuration” mechanism provided by the configuration manager
 - Using features such as locking, loading, browsing, modifying, saving, and committing the configuration
- Accessing the router’s operational data with XML
- Working with the native management data object class hierarchies:
 - To represent the native data objects in XML
 - To use various techniques for structuring XML requests to access the management data of interest, including the use of wildcards and filters
- Encapsulating CLI commands in XML
- How error information is returned to the client application
- Using iterators for large data retrieval
- Handling event notifications with XML
- How authorization of client requests is enforced
- Versioning of the XML schemas
- Generation and packaging of the XML schemas
- Transporting options enable the corresponding XML agents on the router
- How to use the Cisco IOS XR Perl Scripting Toolkit to write Perl scripts to manage a Cisco IOS XR router

Cisco XML API and Router System Features

Using the XML API, an external client application can send XML encoded management requests to an XML agent running on the router. The XML API readily supports available transport layers including the Common Object Request Broker Architecture (CORBA) and the terminal-based protocols Telnet and Secure Shell (SSH). The Cisco XML API interface described in this document applies equally to all supported transports.

Before an XML session is established, the XML transport and XML agent must be enabled on the router. For more information, see [Chapter 12, “XML Transport and Event Notifications.”](#)

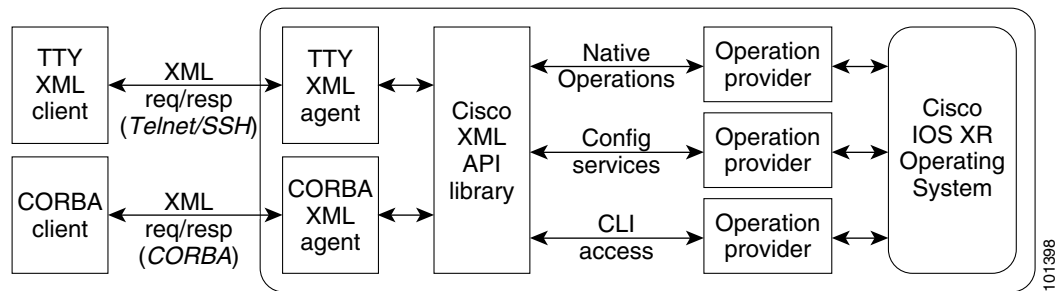
A client request sent to the router must specify the different types of operations to be carried out. The following three general types of management operations are supported through XML:

- Native data access (get, set, delete, and so on) using the native management data model.
- Configuration services for more advanced configuration management through the Configuration Manager.
- Traditional CLI access where the CLI commands and command responses are encapsulated in XML.

When a client request is received by an XML agent on the router, the request is routed to the appropriate XML operation provider in the internal Cisco XML API library for processing. After all the requested operations are processed, the XML agent receives the result and then sends the XML encoded response stream on to the client.

[Figure 1-1](#) presents a high level view of the XML manageability-related components along with the request and response flows between the client application and the router.

Figure 1-1 XML Components and Request/Response Flows



Cisco XML API Tags

An external client application can access management data on the router through an exchange of well structured XML-tagged request and response streams. The XML tagged request and response streams are described in the following sections:

- [Basic XML Request Content, page 1-22](#)
- [XML Declaration Tag, page 1-23](#)
- [Operation Type Tags, page 1-25](#)
- [XML Request Batching, page 1-26](#)

Basic XML Request Content

This section describes the specific content and format of the XML data exchanged between the client and the router for the purpose of router configuration and monitoring.

Top-Level Structure

The top level of every request sent by a client application to the router must begin with an XML declaration tag followed by a request tag and one or more operation type tags. Similarly, every response returned by the router must begin with an XML declaration tag followed by a response tag and one or more operation type tags. Each request can contain operation tags for each supported operation type and the operation type tags can be repeated. The operation type tags contained in the response corresponds to those contained in the client request.

Sample XML Request from Client Application

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

Sample XML Response from Router

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

**Note**

All examples in this document are formatted with new lines and white space to aid readability. Actual XML request and response streams exchanged with the router do not include new lines and white space characters because these elements would add significantly to the size of the XML data and impact the overall performance of the XML API.

XML Declaration Tag

Each request and response exchanged between a client application and the router must begin with an XML declaration tag indicating which version of XML and (optionally) which character set is being used:

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

Table 1-2 defines the attributes of the XML declaration that are defined by the XML specification.

Table 1-2 Attributes for XML Declaration

Name	Description
Version	Specifies the version of XML to be used. Only Version "1.0" is supported by the router. Note The version attribute is required.
Encoding	Specifies the standardized character set to be used. Only "UTF-8" is supported by the router. The router includes the encoding attribute in a response only if it was specified in the corresponding request. Note The encoding attribute is optional.

Request and Response Tags

Following the XML declaration tag a client application must enclose each request stream within a set of `<Request>` start and `</Request>` end tags. Also, the system encloses each XML response within a set of `<Response>` start and `</Response>` end tags. A major and minor version number are carried on the `<Request>` and `<Response>` elements to indicate the overall XML API version in use by the client application and router respectively.

The XML API presents a synchronous interface to client applications. The `<Request>` and `<Response>` tags are used by the client to correlate the request and response streams. A client application can issue a request after which the router returns a response. The client can then issue another request, and so on. Therefore, the XML session between a client and the router consist of a series of alternating requests and response streams.

The client application optionally includes a `ClientID` attribute within the `<Request>` tag. The value of the `ClientID` attribute must be an unsigned 32-bit integer value. If the `<Request>` tag contains a `ClientID` attribute, the router includes the same `ClientID` value in the corresponding `<Response>` tag. The `ClientID` value is treated as opaque data and is ignored by the router.

Maximum Request Size

The maximum size of an XML request or response is determined by the restrictions of the underlying transports. For more information on transport-specific limitations of request and response sizes, see [Chapter 12, “XML Transport and Event Notifications.”](#)

Minimum Response Content

If a `<Get>` request has nothing to return for a particular operation, the router returns the original request and an appropriate empty operation type tag. The minimum response returned by the router in the case of a single operation (for example, `<Get>`, `<Set>`, `<Delete>`) with no result data is shown in the following example.

Sample XML Request from Client Application

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

Sample XML Minimum Response from a Router

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

Operation Type Tags

Following the <Request> tag, the client application must specify the operation types to be carried out by the router. Three general types of operations are supported along with the <GetNext> operation for large responses.

Native Data Operation Tags

Native data operations provide basic access to the native management data model. [Table 1-3](#) describes the native data operation tags.

Table 1-3 Native Data Operation Tags

Native Data Tag	Description
<Get>	Get the value of one or more configuration, operational, or action data items.
<Set>	Create or modify one or more configuration or action data items.
<Delete>	Delete one or more configuration data items.
<GetVersionInfo>	Get the major and minor version numbers for one or more components.

The XML schema definitions for the native data operation type tags are contained in the schema file `native_data_operations.xsd`. The native data operations are described further in [Chapter 5, “Cisco XML and Native Data Access Techniques.”](#)

Configuration Services Operation Tags

Configuration services operations provide more advanced configuration management functions through the Configuration Manager. [Table 1-4](#) describes the configuration services operation tags.

Table 1-4 Configuration Services Operation Tags

Tag	Description
<Lock>	Lock the running configuration.
<Unlock>	Unlock the running configuration.
<Load>	Load the target configuration from a binary file previously saved by way of the <Save> tag.
<Save>	Save the target configuration to a binary file.
<Commit>	Promote the target configuration to the running configuration.
<Clear>	Abort/clear the current target configuration session.
<Rollback>	Roll back the running configuration to a previous configuration state.
<GetConfigurationHistory>	Get a list of commits made to the running configuration.
<GetConfigurationSessions>	Get a list of the user sessions currently configuring the box.

The XML schema definitions for the configuration services operation type tags are contained in the schema file `config_services_operations.xsd` (see [Chapter 14, “Cisco XML Schemas”](#)).

The configuration services operations are described further in [Chapter 2, “Cisco XML Router Configuration and Management.”](#)

CLI Operation Tag

CLI access provides support for XML encapsulated CLI commands and responses. For CLI access, a single tag is provided. The `<CLI>` operation tag issues the request as a CLI command.

The XML schema definitions for the CLI tag are contained in the schema file `cli_operations.xsd` (see [Chapter 14, “Cisco XML Schemas”](#)).

The CLI operations are described further in [Chapter 6, “Cisco XML and Encapsulated CLI Operations.”](#)

GetNext Operation Tag

The `<GetNext>` tag is used to retrieve the next portion of a large response. It can be used as required to retrieve an oversize response following a request using one of the other operation types. The `<GetNext>` operation tag gets the next portion of a response. Iterators are supported for large requests.

The XML schema definition for the `<GetNext>` operation type tag is contained in the schema file `xml_api_protocol.xsd` (see [Chapter 14, “Cisco XML Schemas”](#)). For more information about the get next operation, see [Chapter 7, “Cisco XML and Large Data Retrieval \(Iterators\).”](#)

Alarm Operation Tags

The `<Alarm>` operation tag registers, unregisters, and receives alarm notifications. [Table 1-5](#) lists the alarm operation subtags.

Table 1-5 List of Alarm Operation Subtags

Subtag	Description
<code><Register></code>	Registers to receive alarm notifications.
<code><Unregister></code>	Cancel a previous alarm notification registration.

The XML schema definitions for the alarm operation tags are contained in the schema file `alarm_operations.xsd` (see [Chapter 14, “Cisco XML Schemas”](#)).

For more information about the alarm operations, see “Cisco XML Transport and CORBA IDL” section on page 12-115 of [Chapter 12, “XML Transport and Event Notifications.”](#)

XML Request Batching

The XML interface supports the combining of several requests or operations into a single request. When multiple operations are specified in a single request, the response contains the same operation tags and in the same order as they appeared in the request.

Batched requests are performed as a “best effort.” For example, if operations 1 through 3 are in the request and operation 2 fails, operation 3 is attempted.

If you want to perform two or more <Get> operations, and the first one can return a large amount of data that is potentially larger than the size of one iterator chunk, you must place the subsequent operations within a separate XML request. If the operations are placed in the same request within the same <Get> tags, for example, potentially sharing part of the hierarchies with the first request, an error attribute that informs you that the operations cannot be serviced is returned on the relevant tags.

For more information, see [“XML Request with Combined Object Class Hierarchies”](#) section on page 5-73 of Chapter 5, “Cisco XML and Native Data Access Techniques.”

The following example shows a simple request containing six different operations:

Sample XML Client Batched Requests

```
<?xml version="1.0" encoding="UTF-8"?>
<Request MajorVersion="1" MinorVersion="0">
  <Lock/>
  <Get>
    <Configuration>
      .
      .
      .
      Get operation content goes here
      .
      .
      .
    </Configuration>
  </Get>
  <Set>
    <Configuration>
      .
      .
      .
      Set operation content goes here
      .
      .
      .
    </Configuration>
  </Set>
  <Commit/>
  <Get>
    <Operational>
      .
      .
      .
      Get operation content goes here
      .
      .
      .
    </Operational>
  </Get>
</Unlock/>
</Request>
```

Sample XML Response from the Router

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

