Concepts and Requirements for XML Network Configuration

1 Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026 [RFC2026].

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt
The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

2 Abstract

This document defines basic concepts and describes the requirements for a standard network configuration protocol, which could be used to manage the configuration of networks and networking equipment.

The document also discusses a phased approach to developing an XML-based configuration protocol that could provide tangible benefits in the short term, while working towards an XML-based configuration protocol that meets the full requirements.

3 Copyright Notice

Copyright (C) The Internet Society (2001). All Rights Reserved.

4 Conventions Used In This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

5 Table of Contents
Introduction

This document contains a set of requirements for a standard, XML-based protocol to manage the configuration of networks and networking equipment. These technical requirements have been derived from several sources, including public and private discussions with network operators regarding their requirements, discussions with networking equipment vendors regarding the configuration properties of networking equipment, and my own
opinions on the subject.

These requirements are listed in logical order. No prioritization should be inferred from the structure of this document.

This document also proposes a staged approach to the development of XML configuration technology within the IETF. The early stages will provide tangible benefits to the Internet community, while building towards a protocol that will meet the requirements outlined in this document.

This version of the document is a rough first draft, and is intended to serve as a strawman for a discussion of configuration requirements at an XML Configuration BOF, to be held at IETF54 in Yokohama, Japan in July 2002.

It is understood that this document will require significant modification and enhancement to represent the full set of requirements for an XML configuration protocol. This document should be updated and completed later, if there is sufficient interest to continue this work within the IETF.

7 Terminology

ASN.1        Abstract Syntax Notation One
BER          Basic Encoding Rules
DTD          Document Type Declaration
MIB          Management Information Base
PDU          Protocol Data Unit
SNMP         Simple Network Management Protocol
SOAP         Simple Object Access Protocol
SSH          Secure Shell
XML          Extended Markup Language

[This section will be completed in a later draft. Right now, it is just a placeholder for the acronyms that ought to be defined and have references added.]

Wasserman, Editor Expires May 2002
Requirements for Network Configuration June 2002

8 Components of a Configuration Protocol

A configuration protocol can be broken down into several major components:

- Data Model

  The data model is the set of configuration information that can be accessed via the configuration protocol. For example, the SNMP data model defines a string of 7-bit ASCII octets that contains human-readable contact information for
the owner or operator of a piece of networking equipment (sysContact).

- **Data Modeling Language**

A data modeling language is used to define the data model. For example, the SNMP data modeling language is a subset of ASN.1, defined as SMIv2.

A single data model might be represented in more than one data modeling language. However, data modeling languages do restrict the data types and structure types that can be used to define the data model. So it may be possible to construct a data model in one data modeling language that cannot be fully represented in another data modeling language.

It is a base assumption that an XML configuration protocol will use some form of XML as a data modeling language.

- **Data Representation**

A well-defined data representation is used to pass data between systems in a system-independent fashion. For example, ASN.1 BER encoding is used as the data representation for SNMP.

It is a base assumption that an XML configuration protocol will use XML for data representation.

- **Protocol Operations**

A configuration protocol largely consists of a set of operations that can be performed on configuration data. For instance, SNMP supports the following operations: GET, SET, GET-NEXT, GET-BULK, NOTIFY. Other management protocols support additional operations, such as ADD and DELETE.

- **Protocol Messages**

Protocol operations and their results are communicated between two systems in protocol messages (operation requests, operation responses, etc.). Different types of protocol messages may have different formats, and protocol messages may include configuration data, represented using the data representation.

For example, SNMP PDUs are protocol messages, and there are several defined PDU types.

- **Protocol Transport**

Protocol messages will be transmitted using a set of underlying transport protocols. In the case of SNMP, PDUs are transported over UDP/IP. It is also possible to use a higher-level protocol, such as SSH or BEEP, as a protocol transport.
This document outlines the general requirements for a configuration protocol, and the requirements for each of the components described above.

9 General Requirements

This section contains general requirements for a configuration protocol. These requirements must be met in order for a new configuration protocol to achieve acceptance among operators and vendors.

9.1 Fits Modern Configuration Systems

To be useful to operators, a configuration protocol must fit into modern systems for managing configuration data. These systems typically include a database that contains the canonical configuration information for the entire network. Mechanisms exist to:

- Generate the expected configuration of each piece of networking equipment from the database in a common form.

- Post-process the database output into configuration information that matches the configuration syntax of a piece of networking equipment based on equipment type, vendor, model and version.

- Periodically check the configuration of each piece of networking equipment to make sure that it matches the database. This involves dumping the configuration of the device and comparing it to the processed database output.

- Update the configuration of a piece of networking equipment when the database changes, or when its configuration no longer matches the database. This involves determining the difference between the current and target configurations and constructing commands to move from the current configuration to the target configuration with minimal changes.

- Report configuration changes and updates to operators via a logging mechanism, including full information about the change. There should be no need to perform successive queries to understand what was changed.

This type of system could be simplified and improved by the existence of a widely implemented, standard configuration protocol to manage the configuration of networking equipment. This document discussed the requirements for that protocol.

This type of system places several high-level requirements on a configuration protocol:

1. MUST be possible to query and modify all of the configuration data on a device using a single mechanism.

Today, many network equipment vendors only support full configuration through a proprietary command-line interface.
2. MUST be possible to easily transform database output into a set of configuration commands that the device will understand. It must be possible to configure a replacement device entirely from information held in the database.

3. MUST be possible to query the full configuration of a device in a single transaction, such as the "config dump" support included in the CLIs of most backbone routers.

4. MUST be possible to compare sets of configuration data, and to generate the configuration commands needed to move from one configuration to another, with minimal disruption of state on the running device.

5. MUST include a mechanism to send full information about configuration updates, suitable for inclusion in a log. Many vendors currently use vendor-defined syslog messages for this purpose.

6. MUST include a mechanism to generate detailed reports when a change is made to the configuration.

9.2 Easy to Extend and Implement

As indicated above, a configuration protocol will only be useful if it can be used to manage all of the configuration information on a device.

Today, SNMP is available on many devices, and we have an extensive data model described in the SNMP Management Information Base (MIB). However, SNMP can only be used to manage a subset of the configuration information available on most networking equipment. Vendors develop CLI commands concurrent with the development of new features, and SNMP is implemented later, if at all. Often, configuration data is implemented as read-only in vendor SNMP MIB implementations, even when the standard MIB indicates read-write access.

In discussion with vendors, three reasons emerge why SNMP is not implemented, with full read-write support, for all new features in a timely manner:

- Standard MIBs are not usually available for new protocols when they are standardized by the IETF.

- It is difficult and time-consuming to develop proprietary MIB extensions, because the SNMP data model does not map easily onto the data structures used in most implementations.

- Even when standard MIBs are available, it is difficult and time-consuming to instrument SNMP MIBs, particularly GET-NEXT and SET processing.

So, it is clear that a new configuration protocol will not be
successful unless it meets the following requirements:

1. Easily Extensible Data Model

   It must be so simple to extend the data model that it will be practical to require IETF working groups to define data model extensions for configuration at the same time that they standardize a new protocol, or a new version of a protocol.

   It must also be easy enough to extend the data model that it is practical for vendors to provide proprietary extensions to cover all of the proprietary configuration information in their systems. This should include the ability to add new fields to standard-defined structures, as well as the ability to add new sets of configuration information to the data model.

2. Easy to Instrument

   SNMP was successfully optimized to allow for a very small, simple, stateless agent. However, this required the MIB instrumentation to handle complex transaction-related issues, such as atomicity, locking and transaction processing (test/commit/undo).

   Any new configuration protocol should be optimized to make it easy to instrument data model extensions on networking equipment. To accomplish this, the protocol should deal with as many of the complex transaction-related issues as possible.

Wasserman, Editor       Expires May 2002                          7
Requirements for Network Configuration June 2002

9.3 Easy to Develop Client-Side Tools

   Operators require the freedom to develop their own tools in their favorite text-oriented scripting languages (Perl, awk, sed, etc.). To avoid having to learn things twice, the same interface that is available at the command-line MUST be callable from scripts.

   Operators do not want to run specialized applications on the client-side to access configuration data and process it into a form that is accessible from scripts or the command-line. It must be possible to directly access the configuration protocol using a secure, text-oriented transport, such as SSH.

   These needs impose several technical requirements on a configuration protocol:

   1. Human-Friendly Data Representation

      The data representation used for configuration protocol input and output (queries, responses, help messages, etc.) MUST be human-readable and MUST contain information that is useful to an operator who is using the configuration protocol from the command-line.

   2. Computer-Friendly Data Representation
The data representation MUST include well-defined formatting, to allow for easy parsing by text processors and scripting languages.

3. Textual Data Representation
In order to generate configuration input and to parse configuration output using text-oriented scripting languages, the data representation MUST be in a textual form.

There is interest in using XML as the data representation for a configuration protocol, because XML is a textual data representation that provides a good compromise between a human-friendly representation and a computer-friendly representation, largely achieving both goals.

4. Secure, Text-Oriented Protocol Transport
The configuration protocol MUST be defined to use a secure, text-oriented protocol transport.

9.4 Security

Security is a vitally important attribute for any management protocol. The XML configuration protocol should provide both security and privacy. The requirements in this area include:

1. Protocol operations MUST be authenticated.
2. Protocol operations MUST be protected against replay.
3. Protocol messages MUST be encrypted.
4. Protocol messages MUST be protected against corruption or intentional modification en route.

[Security requirements will be specified in more detail in a future draft, hopefully by someone who understands security better than I do.]

It is not necessary for the configuration protocol, itself, to meet all of the security requirements. It is acceptable for some or all of these requirements to be met by the protocol transport (i.e. TCP could be leveraged to protect against accidental corruption en route, SSH could be used to provide encryption, etc.).

9.5 Internationalization

The data model and data representation for text strings MUST support international character sets. For instance, it should be possible for operators to set the system contact information to include their actual names and addresses.

[The requirements in this area should be improved by someone who understands these issues much better than I do.]
One of the most complex issues surrounding the definition of an XML-based configuration protocol within the IETF will be the data model.

The IETF has considerable effort invested in the SNMP data model, represented in the standard MIBs. It is vital to leverage the SNMP data model for any new configuration protocol, and it would be wise to avoid the definition of an entirely new data model. However, it would be hard to accept the limitations that the SNMP data representation, SMIv2, has imposed on the data model.

Although further work is needed to determine our actual requirements in this area, some requirements are clear in theory, although perhaps mutually exclusive in practice:

1. An IETF-defined data model MUST leverage the existing SNMP data model.

2. The data model MUST make a clear distinction between configuration information and state information, and it MUST be possible to query the full configuration of a device, without retrieving all of the dynamic state information held on the device. For example, it should be possible to query the full configuration of a router, including static routing and forwarding entries, without retrieving all of the dynamic entries in the router’s routing and forwarding tables.

3. The data model MUST support a wider range of types than the SNMP data model. For example, it MUST support international character sets in text strings and SHOULD support structured data types (aggregates of basic data types).

4. The data model MUST be easy to extend. This includes the ability to extend standards-defined data structures and the ability to add new sets of configuration data to the data model.

Fortunately, the earliest stages of the phased approach outlined below will allow us to defer standardization of a data model for configuration. The popularity of CLI-based configuration tools demonstrates that it is possible to have a useful, widely deployed configuration mechanism that does not include a standardized data model.

**11 Data Modeling Language Requirements**

By definition, an XML configuration protocol would use an XML-based data modeling language. Further exploration is required to determine whether it would be better to use an XML schema or a DTD to define configuration data.

There are some basic requirements for a data modeling language:

1. The data modeling language MUST, at a minimum, be capable of
representing the SNMP data model. This include the ability to represent:

A. All SMIv2 data types and textual conventions,
B. Scalar variables, and
C. List of objects (i.e. one-dimensional arrays)

This requirement allows SNMP MIBs to be fully represented in the XML-based data modeling language, so the XML configuration protocol could potentially access all SNMP MIB data.

Some work has already been done on programmatic conversions from SMIv2 to XML, and this work will be discussed at the XML Configuration BOF in Yokohama.

2. In order to conveniently express the data included in most complex programs, the data modeling language MUST support:

A. Basic object types (i.e. integers, and strings)
B. Basic objects with added semantics (i.e. counters)
C. Structure types (aggregates of basic types)
D. Lists of basic objects
E. Lists of structured objects

3. The data modeling language MAY also support more complex data structures than simple aggregates and lists, such as:

A. Complex table or tree representations
B. Singly-linked or doubly-linked lists
C. Lists within lists

4. In order to validate configurations (see below), the data modeling language MUST support the ability to define a function that can be invoked on a piece of networking equipment, including input parameters and return values.

An important issue to resolve will be whether it should also be possible to represent the XML configuration protocol’s data model in SNMP MIBs. This would allow all configuration data to be monitored using SNMP. If required, this generates an additional requirement for the data modeling language:

5. The data modeling language MUST be isomorphic to SMIv2.

Unfortunately, this requirement would severely restrict the capabilities of an XML configuration protocol, and might be mutually incompatible with other requirements, such as the ability to easily extend the data model and represent a wider range of types than can be represented in SMIv2.
Further effort is needed to resolve these trade-offs and to determine the requirements for a data modeling language. However, the early stages of the phase approach described below do not require a standard data modeling language for XML configuration, so it may not be necessary to immediately address these issues. Ongoing work on a next generation SMI (in the SMInt WG) may eliminate some of the restrictions associated with SMIV2.

12 Data Representation Requirements

By definition, an XML-based configuration protocol would use an XML-based data representation. There is no requirement that the data representation for an XML configuration protocol maps, in any way, to the data representation used by other management protocols.

Detailed requirements for the data representation can also be deferred to later stages, if the phased approach is adopted.

13 Protocol Operations Requirements

A set of protocol operations defines the fundamental capabilities of a configuration protocol. Some configuration operations deal with opaque blocks of configuration information, representing whole or partial device configurations. Other configuration operations are performed on specific pieces of configuration data.

The approach described in this document suggests that a useful configuration protocol could be defined that deals only with opaque blocks of configuration information. The actual configuration information could be represented in a form that is vendor-proprietary.

Later work would define protocol operations to deal with specific pieces of configuration data.

13.1 Operations on Configuration Blocks

An XML protocol should support a set of operations that deal with opaque blocks of configuration data. These operations will allow configuration data to be queried and modified, without requiring the protocol to deal with individual pieces of configuration information.

13.1.1 Operations on Whole Configurations

The XML configuration protocol MUST, at a minimum, support two operations on whole configurations:

1. Dump Whole Configuration
   The protocol MUST support a mechanism to dump the entire configuration of a running piece of networking equipment. It MUST be possible to dump configuration information without dumping dynamic state information.

2. Restore Whole Configuration
   The protocol MUST also support the ability to send a full
set of configuration data to a piece of networking equipment. This information could be used to fully configure a system, when first deployed or after a catastrophic failure.

In addition to these two functions, there is a further requirement for the server running on the networking equipment:

3. A piece of networking equipment SHOULD be able to update its running configuration to match a downloaded configuration block, with minimal disruption of state. Essentially, this requires that an XML configuration server include the ability to do a logical "diff" between two configurations, and move from the current configuration to a new configuration by executing the minimal set of configuration changes required.

13.1.2 Operations on Partial Configurations

It may also be useful to support operations on opaque blocks of configuration data that represent partial system configurations. For instance, it might be useful to dump and restore the configuration for a single protocol or sub-system.

Further thought is required to determine how partial configurations could be named and referenced. It might not be meaningful to perform operations on blocks of data without a defined format for configuration blocks. This may be as simple as stating that configuration blocks consist of data in name/type/value triplets.

13.2 Operations on Configuration Data Objects

In addition to operations on blocks of data, the XML configuration protocol will include operations to manipulate individual configuration objects.

There are several basic requirements for operations on data objects:

1. The protocol MUST include operations to access basic objects (integers, strings, etc.).

2. The protocol MUST include operations to access structured objects (aggregates of basic types).

3. The protocol MUST include operations to access lists of objects.

4. Operations MUST be provided to get and set objects.

5. Operations MUST be provided to add and delete basic objects from object lists.

13.3 Validating Configurations
It should be possible to use the XML configuration protocol to validate the configuration of a piece of networking equipment.

This will typically involve invoking a special purpose function on a device and checking the return from that function. For example, it should be possible to verify a forwarding table configuration by calling a forwarding function with an IP address. The return value would indicate the next-hop information that would be used to forward the packet.

Support for configuration validation implies several requirements for an XML configuration protocol:

1. It MUST be possible to identify and invoke a function that will run on the networking equipment.
2. It MUST be possible to pass parameters to the function.
3. It MUST be possible to receive return values from the function.

13.4 Transaction Handling

The transaction handling features of the XML configuration protocol should be carefully considered and well defined. This includes concepts such as locking, atomicity and roll-back.

Transaction processing generally consists of several steps:

- Test: Determines whether or not an operation is valid for each piece of data.
- Perform: Actually performs the operation, saving information that may be required for roll-back.
- Complete: Operation completed successfully, roll-back information can be freed.
- Roll-Back: Return the configuration to its previous state. This step is invoked when an error occurs after an operation is partially completed, or when a timeout occurs before the complete step is invoked.

13.4.1 Single-Device Transactions

An internal transaction occurs within a single device. Externally, a single operation (i.e., a set operation) is invoked on a group of objects. Single-device transaction handling controls locking the affected data structures during the operation, and performing the operation atomically for the group of objects. This includes the ability to roll-back partial changes when an error occurs.

The XML configuration protocol MUST support single-device transaction handling for any operations that modify the configuration of a device. At a minimum, this would include restore, set, add and delete operations.

13.4.2 Multiple-Device Transactions

An external transaction can be performed, in an atomic fashion,
across multiple pieces of networking equipment. For example, it should be possible to renumber the routers at both ends of a point-to-point link in a single, atomic transaction. In order to support external transactions, it must be possible for the client to invoke the individual transaction steps (test, perform, complete and rollback).

The XML configuration protocol SHOULD support multiple-device transaction handling. This should include the ability to roll-back configuration changes when indicated.

14 Protocol Messaging Requirements

This section discusses the requirements for protocol messaging. Protocol messaging is, essential, an RPC mechanism that can be used to exchange operations and data objects (as operation input parameters or as return values). It is quite possible that a currently available protocol, such as SOAP, could be used as the protocol messaging portion of the XML configuration protocol.

14.1 Protocol Message Types

It is likely that an XML configuration protocol will include several message types. Further thought is required to determine what types of protocol messages should be supported.

14.2 Authentication of Protocol Messages

It is vital that protocol messages are both authenticated and authorized before any operations contained within the message are performed. At a minimum, the XML configuration protocol must meet the following requirements:

1. All protocol messages MUST be authenticated and authorized before any protocol operations contained within the message are performed.

2. The XML configuration protocol MAY rely on authentication features of the underlying protocol transport.

14.3 Privacy of Protocol Messages

Configuration messages will often contain private information, such as security keys and user information. The XML configuration protocol is expected to include dump and restore operations that move blocks of human-readable configuration information, without the protocol being aware of the contents. In order to protect sensitive configuration information, the following requirements must be met:

1. All protocol messages that contain opaque blocks of configuration data MUST be encrypted.

2. The protocol MAY include a mechanism to tag private and non-private data.
If a tagging mechanism is included, all protocol messages containing private data MUST be encrypted.

If a tagging mechanism is not included, all protocol messages containing any configuration data MUST be encrypted.

3. The protocol MAY rely on encryption capabilities of the underlying protocol transport to meet the privacy requirements.

15 Protocol Transport Requirements

As indicated in the General Requirements section, the XML configuration protocol should use a secure, text-oriented protocol transport, such as SSH. Requirements for the protocol transport include:

1. It MUST be possible and practical for a human to view and modify configuration information from a command-line.

2. It MUST be possible to view and modify configuration information using scripts written in text-oriented scripting languages.

3. The XML configuration protocol MUST NOT require a specialized application, such as an SNMP MIB browser, on the client-side. The protocol MUST be transported over a widely available, general-purpose protocol, such as SSHv2 or HTTP.

4. The protocol transport SHOULD provide basic security features such as authentication and encryption, to avoid the need to define basic security mechanisms specifically for the XML configuration protocol.

16 Approach to XML Configuration

This section offers thoughts on how the IETF could approach the development of an XML-based configuration protocol. The protocol could be developed in a set of discrete phases that would each offer tangible benefits to the Internet community, while building towards a protocol that would meet the full set of requirements outlined in this document.

This approach would also allow the IETF to commit to a limited set of XML-related work, and to determine the usefulness and industry acceptance of that work before committing to later phases.

A more aggressive plan could combine multiple phases into a single block of work.

16.1 Phase Zero: Investigate Technical Options

During this phase we would explore existing XML-based technologies and determine their applicability to an XML configuration. The
goal would be to emerge from this phase with a basic understanding of the technical approach that we would pursue.

Output from this phase should include a greatly enhanced concepts and requirements document, and a detailed plan for later phases of protocol development.

Until this phase has been completed, we should not begin work on any of the later phases of this effort.

16.2 Phase One: XML over Secure Transport

The first standardization task of a working group focused on XML configuration should be to standardize how XML data can be transmitted over a secure protocol transport. This would include an explanation of how XML should be encapsulated within the secure transport, and the assignment of a new port number for XML transport connections. It might be possible to base this work on current industry practice.

This phase would basically define the protocol transport for later XML configuration work.

16.3 Phase Two: XML Operations on Configuration Blocks

For the second phase, an XML configuration WG should focus on defining how operations on blocks of configuration information, representing whole or partial system configurations, could be transferred over the secure transport defined in the first phase. This would include a basic RPC-like mechanism for specifying operations on configuration blocks.

This phase would involve developing a basic framework for protocol messaging and the definition of some protocol message types. It would also require specification of a set of protocol operations to manipulate configuration blocks, and authentication and authorization mechanisms to secure those operations. A rudimentary data representation would be required, to represent chunks of opaque XML configuration data.

The contents of the configuration blocks would be vendor-proprietary, and no attempt would be made in this phase to define an XML-specific data model, a data modeling language or a full data representation for XML configuration.

It might be possible to base some of the work in this phase on an existing proprietary or standard XML RPC mechanism, such as SOAP.

16.4 Phase Three: XML Operations on Data Objects

In phase three, it would become necessary to define a data model, data modeling language and complete data representation for the XML configuration protocol. In this phase, we would define operations to manipulate individual configuration data objects.

At this phase, the protocol would still be restricted to operations between a management system (i.e. workstation) and a single piece
of networking equipment (i.e. router or switch).

16.5 Phase Four: Multi-System Transactions using XML

Phase four would involve the addition of multi-system transaction support to the XML configuration protocol. After the completion of phase four, it would be possible to use the XML configuration protocol to configure entire networks, not just individual pieces of networking equipment.

17 Security Considerations

[This section will be completed in a future version of this draft.]

18 References

[RFC2026]
S. Bradner, "The Internet Standards Process -- Revision 3", RFC 2026, BCP9, October 1996

[RFC2119]
S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP14, March 1999.

[This section will be completed in a later draft.]

19 Acknowledgements

Ran Atkinson made several useful suggestions that have been incorporated into this document. Andy Bierman, Randy Bush and Ted Goddard provided feedback to a preliminary version of the document.

20 Author’s Contact Information

Comments or questions regarding this document should be sent to:

Margaret Wasserman
Wind River
10 Tara Blvd., Suite 330 Phone: (603) 897-2067
Nashua, NH 03062 USA Email: mrw@windriver.com