Internet Engineering Task Force (IETF)
Request for Comments: 7923
Category: Informational
ISSN: 2070-1721

E. Voit
A. Clemm
A. Gonzalez Prieto
Cisco Systems

June 2016

Requirements for Subscription to YANG Datastores

Abstract

This document provides requirements for a service that allows client applications to subscribe to updates of a YANG datastore. Based on criteria negotiated as part of a subscription, updates will be pushed to targeted recipients. Such a capability eliminates the need for periodic polling of YANG datastores by applications and fills a functional gap in existing YANG transports (i.e., Network Configuration Protocol (NETCONF) and RESTCONF). Such a service can be summarized as a "pub/sub" service for YANG datastore updates. Beyond a set of basic requirements for the service, various refinements are addressed. These refinements include: periodicity of object updates, filtering out of objects underneath a requested a subtree, and delivery QoS guarantees.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7923.

Table of Contents

1. Introduction 3
2. Business Drivers 3
   2.1. Pub/Sub in the Interface to the Routing System (I2RS) 4
   2.2. Pub/Sub Variants on Network Elements 5
   2.3. Existing Generalized Pub/Sub Implementations 6
3. Terminology 6
4. Requirements 7
   4.1. Assumptions for Subscriber Behavior 7
   4.2. Subscription Service Requirements 8
        4.2.1. General 8
        4.2.2. Negotiation 9
        4.2.3. Update Distribution 10
        4.2.4. Transport 11
        4.2.5. Security Requirements 11
        4.2.6. Subscription QoS 13
        4.2.7. Filtering 14
        4.2.8. Assurance and Monitoring 15
5. Security Considerations 15
6. References 16
   6.1. Normative References 16
   6.2. Informative References 16
Acknowledgments 17
Authors' Addresses 18

1. Introduction

Applications interacting with YANG datastores require capabilities beyond the traditional client-server configuration of network elements. One class of such applications are service-assurance applications, which must maintain a continuous view of operational data and state. Another class of applications are security applications, which must continuously track changes made upon network elements to ensure compliance with corporate policy.

Periodic fetching of data is not an adequate solution for applications requiring frequent or prompt updates of remote object state. Applying polling-based solutions here imposes a load on networks, devices, and applications. Additionally, polling solutions are brittle in the face of communication glitches, and have limitations in their ability to synchronize and calibrate retrieval intervals across a network. These limitations can be addressed by including generic object subscription mechanisms within network elements, and allowing these mechanisms to be applied in the context of data that is conceptually contained in YANG datastores.

This document aggregates requirements for such subscription from a variety of deployment scenarios.

2. Business Drivers

For decades, information delivery of current network state has been accomplished either by fetching from operations interfaces, or via dedicated, customized networking protocols. With the growth of centralized orchestration infrastructures, imperative policy distribution, and YANG's ascent as the dominant data modeling language for use in programmatic interfaces to network elements, this mixture of fetch plus custom networking protocols is no longer sufficient. What is needed is a push mechanism that is able to deliver object changes as they happen.

These push distribution mechanisms will not replace existing networking protocols. Instead they will supplement these protocols, providing different response time, peering, scale, and security characteristics.

Push solutions will not displace all existing operations infrastructure needs. And SNMP and MIBs will remain widely deployed and the de facto choice for many monitoring solutions. But some functions could be displaced. Arguably the biggest shortcoming of SNMP for those applications concerns the need to rely on periodic polling, because it introduces an additional load on the network and devices, because it is brittle if polling cycles are missed, and

because it is hard to synchronize and calibrate across a network. If applications can only use polling type interaction patterns with YANG datastores, similar issues can be expected.

2.1. Pub/Sub in the Interface to the Routing System (I2RS)

Various documents about the Interface to the Routing System (I2RS) highlight the need to provide pub/sub capabilities between network elements. From [RFC7921], there are references throughout the document beginning in Section 6.2. Some specific examples include:

- Section 7.6 of [RFC7921] provides high-level pub/sub (notification) guidance.
- Section 6.4.2 of [RFC7921] identifies "subscribing to an information stream of route changes" and "receiving notifications about peers coming up or going down".
- Section 6.3 of [RFC7921] notes that when Local Configuration preempts I2RS, external notification might be necessary.

In addition, [USECASE] has relevant requirements. A small subset includes:

- L-Data-REQ-12: The I2RS interface should support user subscriptions to data with the following parameters: push of data synchronously or asynchronously via registered subscriptions...
- L-DATA-REQ-07: The I2RS interface (protocol and instant messages (IMs)) should allow a subscriber to select portions of the data model.
- PI-REQ01: Monitor the available routes installed in the Routing Information Base (RIB) of each forwarding device, including near real-time notification of route installation and removal.
- BGP-REQ10: The I2RS client SHOULD be able to instruct the I2RS agent(s) to notify the I2RS client when the BGP processes on an associated routing system observe a route change to a specific set of IP Prefixes and associated prefixes.... The I2RS agent should be able to notify the client via the publish or subscribe mechanism.
- IGP-REQ-07: The I2RS interface (protocol and IMs) should support a mechanism where the I2RS Clients can subscribe to the I2RS Agent's notification of critical node IGP events.

- MPLS-LDP-REQ-03: The I2RS Agent notifications should allow an I2RS client to subscribe to a stream of state changes regarding the LDP sessions or LDP Label Switched Paths (LSPs) from the I2RS Agent.

- L-Data-REQ-01: I2RS must be able to collect large data sets from the network with high frequency and resolution, and with minimal impact to the device's CPU and memory.

Also, Section 7.4.3 of [RFC7922] includes this pub/sub requirement:

- I2RS agents MUST support publishing I2RS trace log information to that feed as described in [this document]. Subscribers would then receive a live stream of I2RS interactions in trace log format and could flexibly choose to do a number of things with the log messages.

2.2. Pub/Sub Variants on Network Elements

This document is intended to cover requirements beyond I2RS. Looking at history, there are many examples of switching and routing protocols that have done explicit or implicit pub/sub in the past. In addition, new policy notification mechanisms that operate on switches and routers are being specified now. A small subset of current and past subscription mechanisms includes:

- Multicast topology establishment is accomplished before any content delivery is made to endpoints (IGMP, PIM, etc.).

- Secure Automation and Continuous Monitoring (SACM) allows subscription into devices, which may then push spontaneous changes in their configured hardware and software [SACMREQ].

- In MPLS VPNs [RFC6513], a Customer Edge router exchanges PIM control messages before Provider Edge (PE) Routing Adjacencies are passed [RFC6513].

- After OSPF establishes its adjacencies, Link State Advertisement will then commence [RFC2328].

Worthy of note in the examples above is the wide variety of underlying transports. A generalized pub/sub mechanism, therefore should be structured to support alternative transports. Based on current I2RS requirements, NETCONF should be the initially supported transport due to the need for connection-oriented/unicast communication. Eventual support for multicast and broadcast subscription update distribution will be needed as well.

2.3. Existing Generalized Pub/Sub Implementations

TIBCO, RSS, Common Object Request Broker Architecture (CORBA), and other technologies all show precursor pub/sub technologies. However, there are new needs (described in Section 4 below) that these technologies do not serve. We need a new pub/sub technology.

There are at least two widely deployed generalized pub/sub implementations that come close to current needs: Extensible Messaging and Presence Protocol (XMPP) [XEP-0060] and Data Distribution Service (DDS) [OMG-DDS]. Both serve as proof-points that a highly scalable distributed datastore implementation connecting millions of edge devices is possible.

Because of these proof-points, we can be comfortable that the underlying technologies can enable reusable generalized YANG object distribution. Analysis will need to fully dimension the speed and scale of such object distribution for various subtree sizes and transport types.

3. Terminology

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 [RFC2119]. Although this document is not a protocol specification, the use of this language clarifies the instructions to protocol designers producing solutions that satisfy the requirements set out in this document.

A Subscriber makes requests for set(s) of YANG object data.

A Publisher is responsible for distributing subscribed YANG object data per the terms of a subscription. In general, a Publisher is the owner of the YANG datastore that is subjected to the subscription.

A Receiver is the target to which a Publisher pushes updates. In general, the Receiver and Subscriber will be the same entity. A Subscription Service provides subscriptions to Subscribers of YANG data.

A Subscription Service interacts with the Publisher of the YANG data as needed to provide the data per the terms of the subscription.

A subscription request for one or more YANG subtrees (including single leafs) is made by the Subscriber of a Publisher and is targeted to a Receiver. A subscription may include constraints that dictate how often or under what conditions YANG information updates might be sent.

A subscription is a contract between a Subscription Service and a Subscriber that stipulates the data to be pushed and the associated terms.

A datastore is defined in [RFC6241].

An Update provides object changes that have occurred within subscribed YANG subtree(s). An Update must include the current status of (data) node instances for which filtering has indicated they have different status than previously provided. An Update may include a bundled set of ordered/sequential changes for a given object that have been made since the last update.

A Filter contains evaluation criteria, which are evaluated against YANG object(s) within a subscription. There are two types of Filters: Subtree Filters, which identify selected objects/nodes published under a target data node, and object element and attribute Filters where an object should only be published if it has properties meeting specified Filter criteria.

4. Requirements

Many of the requirements within this section have been adapted from the XMPP [XEP-0060] and DDS [OMG-DDS] requirements specifications.

4.1. Assumptions for Subscriber Behavior

This document provides requirements for the Subscription Service. It does not define all the requirements for the Subscriber/Receiver. However in order to frame the desired behavior of the Subscription Service, it is important to specify key input constraints.

A Subscriber SHOULD avoid attempting to establish multiple subscriptions pertaining to the same information, i.e., referring to the same datastore YANG subtrees.

A Subscriber MAY provide subscription QoS criteria to the Subscription Service; if the Subscription Service is unable to meet those criteria, the subscription SHOULD NOT be established.

When a Subscriber and Receiver are the same entity and the transport session is lost/terminated, the Subscriber MUST re-establish any subscriptions it previously created via signaling over the transport session. That is, there is no requirement for the life span of such signaled subscriptions to extend beyond the life span of the transport session.

A Subscriber MUST be able to infer when a Subscription Service is no longer active and when no more updates are being sent.

A Subscriber MAY check with a Subscription Service to validate the existence and monitored subtrees of a subscription.

A Subscriber MUST be able to periodically lease and extend the lease of a subscription from a Subscription Service.

4.2. Subscription Service Requirements

4.2.1. General

A Subscription Service MUST support the ability to create, renew, time out, and terminate a subscription.

A Subscription Service MUST be able to support and independently track multiple subscription requests by the same Subscriber.

A Subscription Service MUST be able to support an add/change/delete of subscriptions to multiple YANG subtrees as part of the same subscription request.

A Subscription Service MUST support subscriptions against operational datastores, configuration datastores, or both.

A Subscription Service MUST be able support filtering so that the subscribed updates under a target node might publish only operational data, only configuration data, or both.

A subscription MAY include Filters as defined within a subscription request, therefore the Subscription Service MUST publish only data nodes that meet the Filter criteria within a subscription.

A Subscription Service MUST support the ability to subscribe to periodic updates. The subscription period MUST be configurable as part of the subscription request.

A Subscription Service SHOULD support the ability to subscribe to updates on-change, i.e., whenever values of subscribed data objects change.

For on-change updates, the Subscription Service MUST support a dampening period that needs to be passed before the first or subsequent on-change updates are sent. The dampening period SHOULD be configurable as part of the subscription request.

A Subscription Service MUST allow subscriptions to be monitored. Specifically, a Subscription Service MUST at a minimum maintain information about which subscriptions are being serviced, the terms of those subscriptions (e.g., what data is being subscribed, associated Filters, update policy -- on change, periodic), and the overall status of the subscription -- e.g., active or suspended.

A Subscription Service MUST support the termination of a subscription when requested by the Subscriber.

A Subscription Service SHOULD support the ability to suspend and to resume a subscription on request of a client.

A Subscription Service MAY at its discretion revoke or suspend an existing subscription. Reasons may include transitory resource limitation, credential expiry, failure to reconfirm a subscription, loss of connectivity with the Receiver, operator command-line interface (CLI), and/or others. When this occurs, the Subscription Service MUST notify the Subscriber and update the subscription status.

A Subscription Service MAY offer the ability to modify a subscription Filter. If such an ability is offered, the service MUST provide subscribers with an indication telling at what point the modified subscription goes into effect.

4.2.2. Negotiation

A Subscription Service MUST be able to negotiate the following terms of a subscription:

- The policy, i.e., whether updates are on-change or periodic
- The interval, for periodic publication policy
- The on-change policy dampening period (if the on-change policy is supported)
- Any Filters associated with a subtree subscription

A Subscription Service SHOULD be able to negotiate QoS criteria for a subscription. Examples of subscription QoS criteria may include reliability of the Subscription Service, reaction time between a monitored YANG subtree/object change and a corresponding notification push, and the Subscription Service's ability to support certain levels of object liveliness.

In cases where a subscription request cannot be fulfilled due to insufficient platform resources, the Subscription Service SHOULD include within its decline hints on criteria that would have been acceptable when the subscription request was made. For example, if periodic updates were requested with update intervals that were too short for the specified data set, an alternative acceptable interval period might be returned from the Publisher. If on-change updates were requested with too aggressive a dampening period, then an acceptable dampening period may be returned, or alternatively an indication that only periodic updates are supported for the requested object(s).

4.2.3. Update Distribution

For on-change updates, the Subscription Service MUST only send deltas to the object data for which a change occurred. (Otherwise the subscriber might not know what has actually undergone change.) The updates for each object MUST include an indication of whether it was removed, added, or changed.

When a Subscription Service is not able to send updates per its subscription contract, the subscription MUST notify subscribers and put the subscription into a state indicating that the subscription was suspended by the service. When able to resume service, subscribers need to be notified as well. If unable to resume service, the Subscription Service MAY terminate the subscription and notify Subscribers accordingly.

When a subscription with on-change updates is suspended and then resumed, the first update SHOULD include updates of any changes that occurred while the subscription was suspended, with the current value. The Subscription Service MUST provide a clear indication when this capability is not supported (because in this case, a client application may have to synchronize state separately).

Multiple objects being pushed to a Subscriber, perhaps from different subscriptions, SHOULD be bundled together into a single Update.

The sending of an Update MUST NOT be delayed beyond the Push Latency of any enclosed object changes.

The sending of an Update MUST NOT be delayed beyond the dampening period of any enclosed object changes.

The sending of an Update MUST NOT occur before the dampening period expires for any enclosed object changes.

A Subscription Service MAY, as an option, support a replay capability so that a set of updates generated during a previous time internal can be sent to a Receiver.

4.2.4. Transport

It is possible for updates coming from a Subscription Service to be pushed over different types of transports such as NETCONF, RESTCONF, and HTTP. Beyond existing transports, this Subscription Service will be applicable for emerging protocols such as those being defined in [USECASE]. The need for such transport flexibility drives the following requirements:

- A Subscription Service SHOULD support different transports.
- A Subscription Service SHOULD support different encodings of a payload.
- It MUST be possible for Receivers to associate the update with a specific subscription.
- In the case of connection-oriented transport, when a transport connection drops, the associated subscription SHOULD be terminated. It is up the Subscriber to request a new subscription.

4.2.5. Security Requirements

Some uses of this Subscription Service will push privacy-sensitive updates and metadata. For privacy-sensitive deployments, subscription information MUST be bound within secure, encrypted transport-layer mechanisms. For example, if NETCONF is used as transport, then [RFC7589] would be a valid option to secure the transported information. The Subscription Service can also be used with emerging privacy-sensitive deployment contexts as well. As an example, deployments based on [USECASE] would apply these requirements in conjunction with those documented within [I2RS-ENV-SEC] and [I2RS-PROT-SEC] to secure ephemeral state information being pushed from a network element.

As part of the subscription establishment, mutual authentication MUST be used between the Subscriber and the Subscription Service.

Subscribers MUST NOT be able to pose as the original Subscription Service.

Versioning of any subscription protocols MUST be supported so that the capabilities and behaviors expected of specific technology implementations can be exposed.

A subscription could be used to attempt to retrieve information to which a client has no authorized access. Therefore, it is important that data being pushed based on subscriptions is authorized in the same way that regular data retrieval operations are authorized. Data being pushed to a client MUST be filtered accordingly, just like if the data were being retrieved on demand. For Unicast transports, the NETCONF Authorization Control Model applies.

Additions or changes within a subscribed subtree structure MUST be validated against authorization methods before subscription updates, including new subtree information, are pushed.

A loss of authenticated access to the target subtree or node SHOULD be communicated to the Subscriber.

For any encrypted information exchanges, commensurate strength security mechanisms MUST be available and SHOULD be used. This includes all stages of the subscription and update push process.

Subscription requests, including requests to create, terminate, suspend, and resume subscriptions MUST be properly authorized.

When the Subscriber and Receiver are different, the Receiver MUST be able to terminate any subscription to it where objects are being delivered over a Unicast transport.

A Subscription Service SHOULD decline a subscription request if it is likely to deplete its resources. It is preferable to decline a subscription when originally requested, rather than having to terminate it prematurely later.

When the Subscriber and Receiver are different, and when the underlying transport connection passes credentials as part of transport establishment, then potentially pushed objects MUST be excluded from a push update if that object doesn't have read access visibility for that Receiver.

4.2.6. Subscription QoS

A Subscription Service SHOULD be able to negotiate the following subscription QoS parameters with a Subscriber: Dampening, Reliability, Deadline, and Bundling.

A Subscription Service SHOULD be able to interpret subscription QoS parameters, and only establish a subscription if it is possible to meet the QoS needs of the provided QoS parameters.

4.2.6.1. Liveliness

A Subscription Service MUST be able to respond to requests to verify the Liveliness of a subscription.

A Subscription Service MUST be able to report the currently monitored Nodes of a subscription.

4.2.6.2. Dampening

A Subscription Service MUST be able to negotiate the minimum time separation since the previous update before transmitting a subsequent update for subscription. (Note: this is intended to confine the visibility of volatility into something digestible by the receiver.)

4.2.6.3. Reliability

A Subscription Service MAY send Updates over Best Effort and Reliable transports.

4.2.6.4. Coherence

For a particular subscription, every update to a subscribed object MUST be sent to the Receiver in sequential order.

4.2.6.5. Presentation

The Subscription Service MAY have the ability to bundle a set of discrete object notifications into a single publishable update for a subscription. A bundle MAY include information on different Data Nodes and/or multiple updates about a single Data Node.

For any bundled updates, the Subscription Service MUST provide information for a Receiver to reconstruct the order and timing of updates.

4.2.6.6. Deadline

The Subscription Service MUST be able to push updates at a regular cadence that corresponds with the Subscriber's specified start and end timestamps. (Note: the regular cadence can drive one update, a discrete quantity of updates, or an unbounded set of periodic updates.)

4.2.6.7. Push Latency

The Subscription Service SHOULD be able to delay Updates on object push for a configurable period per Subscriber.

It MUST be possible for an administrative entity to determine the Push latency between object change in a monitored subtree and the Subscription Service Push of the update transmission.

4.2.6.8. Relative Priority

The Subscription Service SHOULD support the relative prioritization of subscriptions so that the dequeuing and discarding of push updates can consider this if there is insufficient bandwidth between the Publisher and the Receiver.

4.2.7. Filtering

If no filtering criteria are provided, or if filtering criteria are met, updates for a subscribed object MUST be pushed, subject to the QoS limits established for the subscription.

It MUST be possible for the Subscription Service to receive Filter(s) from a Subscriber and apply them to the corresponding object(s) within a subscription.

It MUST be possible to attach one or more Subtree and/or object element and attribute Filters to a subscription. Mandatory Filter types include:

- For character-based object properties, Filter values that are exactly equal to a provided string, not equal to the string, or containing a string.

- For numeric object properties, Filter values that are =, !=, <, <=, >, or >= a provided number.

It SHOULD be possible for Filtering criteria to evaluate more than one property of a particular subscribed object as well as apply multiple Filters against a single object.

It SHOULD be possible to establish query match criteria on additional objects to be used in conjunction with Filtering criteria on a subscribed object. (For example, if A has changed and B=1, then Push A.) Query match capability may be done on objects within the datastore even if those objects are not included within the subscription. This of course assumes that the subscriber has read access to those objects.

For on-change subscription updates, an object MUST pass a Filter through a Filter if it has changed since the previous update. This includes if the object has changed multiple times since the last update, and if the value happens to be the exact same value as the last one sent.

4.2.8. Assurance and Monitoring

It MUST be possible to fetch the state of a single subscription from a Subscription Service.

It MUST be possible to fetch the state of all subscriptions of a particular Subscriber.

It MUST be possible to fetch a list and status of all subscription requests over a period of time. If there is a failure, some failure reasons might include:

- Improper security credentials provided to access the target node;
- Target node referenced does not exist;
- Subscription type requested is not available upon the target node;
- Out of resources, or resources not available;
- Incomplete negotiations with the Subscriber.

5. Security Considerations

There are no additional security considerations beyond the requirements listed in Section 4.2.5.

6. References

6.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <http://www.rfc-editor.org/info/rfc2328>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6513] Rosen, E., Ed. and R. Aggarwal, Ed., "Multicast in MPLS/BGP IP VPNs", RFC 6513, DOI 10.17487/RFC6513, February 2012, <http://www.rfc-editor.org/info/rfc6513>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <http://www.rfc-editor.org/info/rfc7589>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <http://www.rfc-editor.org/info/rfc7921>.

[RFC7922] Clarke, J., Salgueiro, G., and C. Pignataro, "Interface to the Routing System (I2RS) Traceability: Framework and Information Model", RFC 7922, DOI 10.17487/RFC7922, June 2016, <http://www.rfc-editor.org/info/rfc7922>.

6.2. Informative References

[I2RS-ENV-SEC]
Migault, D., Ed., Halpern, J., and S. Hares, "I2RS Environment Security Requirements", Work in Progress, draft-ietf-i2rs-security-environment-reqs-01, April 2016.

[I2RS-PROT-SEC]
Hares, S., Migault, D., and J. Halpern, "I2RS Security Related Requirements", Work in Progress, draft-ietf-i2rs-protocol-security-requirements-06, May 2016.

[OMG-DDS] Object Management Group (OMG), "Data Distribution Service for Real-time Systems, Version 1.2", January 2007, <http://www.omg.org/spec/DDS/1.2/>.

[SACMREQ] Nancy, N. and L. Lorenzin, "Security Automation and Continuous Monitoring (SACM) Requirements", Work in Progress, draft-ietf-sacm-requirements-13, March 2016.

[USECASE] Hares, S. and M. Chen, "Summary of I2RS Use Case Requirements", Work in Progress, draft-ietf-i2rs-usecase-reqs-summary-02, March 2016.

[XEP-0060] Millard, P., Saint-Andre, P., and R. Meijer, "Publish-Subscribe", XSF XEP-0060, July 2010, <http://xmpp.org/extensions/xep-0060.html>.

Acknowledgments

We wish to acknowledge the helpful contributions, comments, and suggestions that were received from Ambika Tripathy and Prabhakara Yellai as well as the helpfulness of related end-to-end system context info from Nancy Cam Winget, Ken Beck, and David McGrew.

Authors' Addresses

Eric Voit
Cisco Systems

Email: evoit@cisco.com

Alexander Clemm
Cisco Systems

Email: alex@cisco.com

Alberto Gonzalez Prieto
Cisco Systems

Email: albertgo@cisco.com