Workgroup: Network Working Group
Internet-Draft: draft-bormann-t2trg-stp-03
Published: 8 April 2020
Intended Status: Informational
Expires: 10 October 2020
Authors: C. Bormann K. Hartke
Universität Bremen TZI Ericsson

The Series Transfer Pattern (STP)

Abstract

Many applications make use of Series of data items, i.e., an array of data items where new items can be added over time. Where such Series are to be made available using REST protocols such as CoAP or HTTP, the Series has to be mapped into a structure of one or more resources and a protocol for a client to obtain the Series and to learn about new items.

Various protocols have been standardized that make Series-shaped data available, with rather different properties and objectives. The present document is an attempt to extract a common underlying pattern and to define media types and an access scheme that can be used right away for further protocols that provide Series-shaped data.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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."

This Internet-Draft will expire on 10 October 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

(TO DO: Insert an extended form of the abstract first here, expanding the reference to [RFC7230] and [RFC7252] in the process.)

Examples for protocols that provide Series-shaped data are:

* The Atom Syndication Format [RFC4287] defines *feeds* as Series of *entries* (links plus some metadata, which can often be much of the content of an entry), where a feed is represented by a collection resource that contains just a small number of the most recent entries. By polling a feed, a client can contain a fresh view of the Series, with a focus on recent items. If the client does not poll often enough, it will *miss* items.

* Messaging protocols such as XMPP or SIMPLE transfer series of what is often called "Instant Messages". A publish/subscribe

mechanism allows a client to select sequences of messages that it is interested in.

* Mail servers that provide interactive access to stored messages present a Series to their clients. Obviously, loss of messages is frowned upon.

* CoAP Observe allows a client to observe a resource as it changes; the client can collect the changes into a Series. Observe is focused on eventual consistency: a fresher data items simply overwrites an older one. The present document uses the observe pattern to build a more general Series Transfer Pattern.

* Syslog is an interesting case of a Series Transfer.

* [RFC8641], [I-D.voit-netmod-yang-notifications2], [RFC8639], [I-D.ietf-netconf-notification-messages], [RFC8650].

* An RTP stream can be viewed as an (somewhat extreme) case of a Series; new items are just sent inside separate UDP packets. A sequence number allows to detect (but not normally ask for retransmission of) missing items. A timestamp as well as source data (SSRC, CSRC) provide further common metadata that aid in the processing of the Series items.

* An example of an ad-hoc design of a series transfer protocol is [I-D.ietf-netconf-udp-pub-channel].

* Server-sent events [sse] are a somewhat bizarre version of a series transfer protocol.

* The Interface for Metadata Access Points (IF-MAP) specified by the Trusted Computing Group and emerging derivatives of that protocol create a series of updates to a graph representation of related network-related security information. The requests created by IF-MAP clients are bundled operations of updates to a MAP Graph, which compose a Series Transfer Pattern of bundled atomic operations that ensure the integrity of the MAP Graph. [Henk Birkholz]

* netflow/IPFIX was defined to transfer a series of data items about flows. Information about PDU flows accounted by network interfaces of endpoints is emitted in a series of counter bundles via the IPFIX protocol. Only a series of these continuous Flow Records creates a meaningful bigger picture about the current traffic in the network topology of an administrative domain. Depending on the characteristics measured, loss of a Flow Record can range from harmless to missing the only vital counter measurement. [Henk Birkholz]

* TO DO: Add more items.

[I-D.birkholz-yang-push-coap-problemstatement] is a problem statement that will require the design of another scheme to transfer Series-shaped data.

2. Objectives

Series transfer applications may have rather different objectives.

* The completeness of the Series transfer may be of utmost importance (e.g., if each item represents a sale), it may be desirable but can be jettisoned in an overload situation, or it may just be a likely outcome with a very active client (e.g., with Atom). Note that there is never a way to *guarantee* completeness unless all of the rate and size of incoming new items, the transfer capacity available, and the processing capabilities of the client are controlled; however, system designs may want to give the illusion of "reliability".

* Minimizing the latency of the transfer may be important, as may be limiting it below a defined maximum (note that these are different objectives). The latter can be supported in a polling system by polling at least as often as that maximum latency; this may be considered inefficient and "push" mechanisms may be developed. Mail environments have developed "push" services to enable minimizing the latency. Where latency requirements go below the time that might be needed for an end-to-end retransmission, error concealment may provide an acceptable user experience (e.g., in RTP).

In general, minimizing latency and ensuring completeness are competing objectives.

Series transfer environments sometimes centralize information distribution functions, leading to "broker" architectures (often combined with the "publish/subscribe" pattern). With brokers, Series publishers may use an entirely different interface to the brokers from that used by the receiving clients, or the interfaces can be designed so they are similar for all the forwarding steps.

3. A REST Series Transfer Pattern (STP)

3.1. Basic collections

A series of items can be represented by a single collection resource:

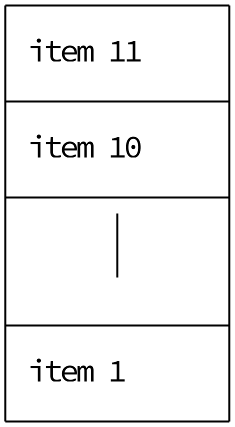

Figure 1: A collection of items

While this is adequate in many cases, it has a number of limitations:

- *Each retrieval fetches the entire collection
  - -As long as the collection does not change, this can be mitigated with ETags (Section 5.10.6 of[RFC7252], Section 2.3 of [RFC7232]).
- *When the collection becomes too large, the server has to prune older items. These then no longer can be retrieved, and there is even no way for the server to indicate that there used to be older items.

3.2. Pagination and Observing linked lists

In the Browser Web, it is usual to provide *Pagination* for collection resources that can grow large (e.g., search results):

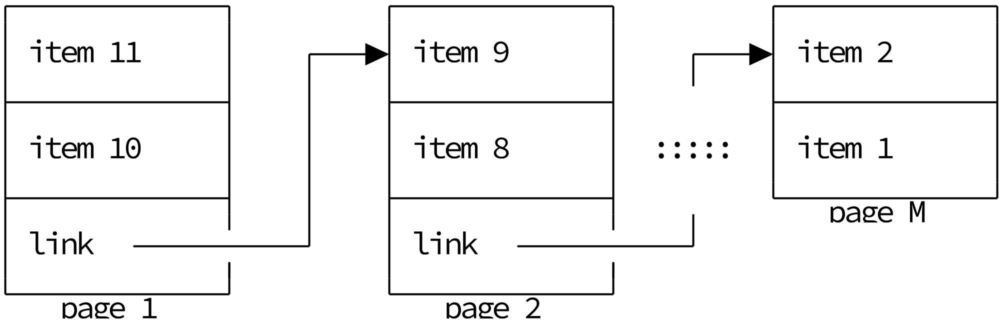

Figure 2: A paginated collection of items

Without modification, this does not work well for resources that actually change by themselves: Once a new page needs to be added, what previously was page 1 now becomes page 2. Obviously, the naming of pages better remains unchanged with new pages added a the front.

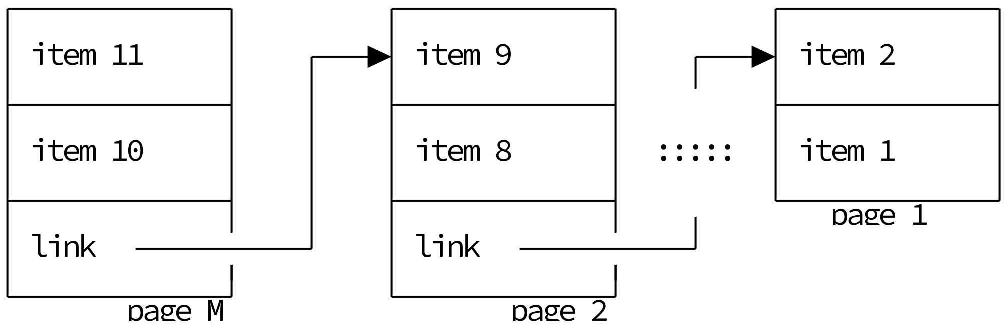

Figure 3: Pagination with stable names

However, now the client has no idea what initial page to request to get the freshest items and the head of the list. It is easy to add a link to the freshest page:

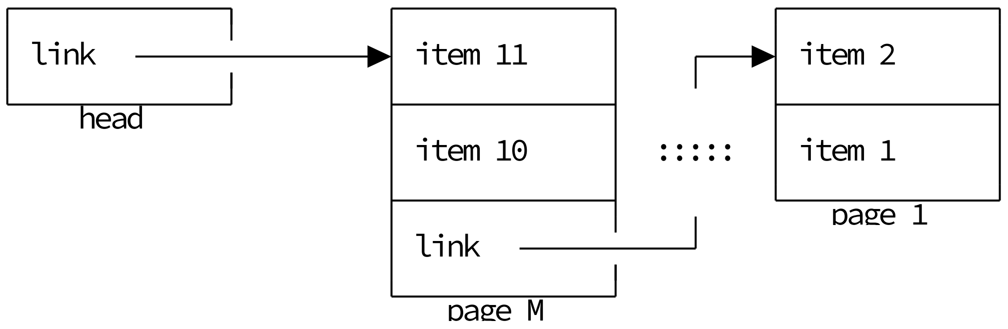

Figure 4: Pagination with stable names

The head of the linked list can now be simply observed; the addition of pages will then be notified to the observer.

As usual in series transfer, the following considerations remain:

* When can the server decide to no longer retain older items?

  - There may be a desire for an observer to be able to catch all items in the series.

    o How does the server know who are the observers? E.g., what to do with newly joining observers?

    o How does an observer signal that it has caught up (to a specific item)?

* What to do when the decision to remove items from the list cannot be made and there is no room for new items?

The link head can also include items that have so far not been added to pages; this can be used to fill up pages evenly without them ever changing. Obviously, the best number of items to prenotify in this way as well as the best time to open a new page are different for different applications.

3.3. The "cursor" pattern

A GET on a resource representing a Series may return a collection item that contains the following pieces of information

* An array of Series items, either as an array of media-typed objects in a suitable representation format (e.g., CBOR, MIME) or by using an array-like media type (e.g., SenML).

  - Items may be full items or limit themselves to some metadata and a link; the client can then follow that link if it is interested in the data (possibly basing that decision on the metadata and/or a measure of load).

* A "cursor" that can then be used as a parameter in further GET requests (see below) in order to receive only newer items than those received with this transfer.

* A "more bit" that indicates whether such further items already exist but could not be returned in the present response.

In Figure 5, the cursor is implemented as a URI that can be used as a link to the next page.

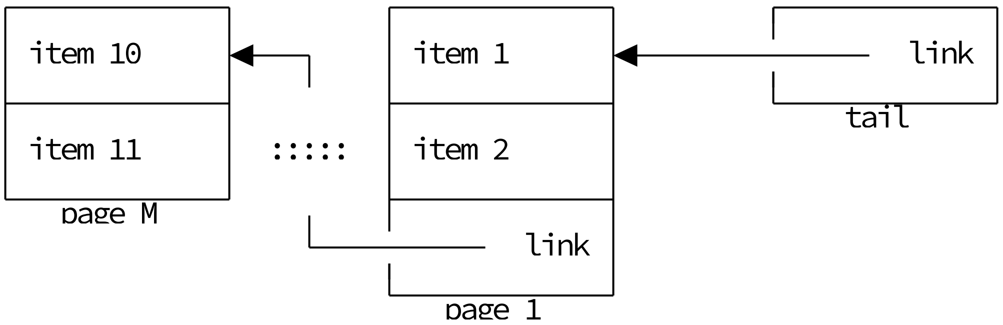

Figure 5: Cursor pattern pictured as pagination

A GET may be enhanced with additional parameters (possibly turning it into a FETCH):

* The cursor.

* A "wait bit" that indicates whether a (possibly empty) reply should be given right away or the server should wait for new items to become available. (To avoid the equivalence of the "silly window syndrome", the wait bit may be enhanced by a minimum number of items and a timeout after which even a smaller number is made available.) In effect, this requests a form of "long polling"; see [RFC6202] for some considerations for this in HTTP.

A server may implement a form of custody transfer by interpreting the cursor as an acknowledgement that the client has received all data up to the cursor. This is not necessarily acting as an unsafe request ("destructive GET"), as other clients may be active that have not yet received all these data. To implement a full custody semantics, the server needs to be aware of all the clients that expect a full Series Transfer (a classical group management problem).

(Explain how Observe can help. Can it?)

4. IANA considerations

This memo registers a number of media types: TO DO.

* A media type for FETCH selectors (Section 3):

  - An alternative way to encode this information into the URI of a GET should also be available.

* A Series media type as alluded to in Section 3.

5. Security considerations

TO DO

6. Informative References

[I-D.birkholz-yang-push-coap-problemstatement]
Birkholz, H., Zhou, T., Liu, X., and E. Voit, "YANG Push Operations for CoMI", Work in Progress, Internet-Draft, draft-birkholz-yang-push-coap-problemstatement-00, 18 October 2017, <http://www.ietf.org/internet-drafts/draft-birkholz-yang-push-coap-problemstatement-00.txt>.

[I-D.ietf-netconf-notification-messages]
Voit, E., Jenkins, T., Birkholz, H., Bierman, A., and A. Clemm, "Notification Message Headers and Bundles", Work in Progress, Internet-Draft, draft-ietf-netconf-notification-messages-08, 17 November 2019, <http://www.ietf.org/internet-drafts/draft-ietf-netconf-notification-messages-08.txt>.

[I-D.ietf-netconf-udp-pub-channel]
Zheng, G., Zhou, T., and A. Clemm, "UDP based Publication Channel for Streaming Telemetry", Work in Progress, Internet-Draft, draft-ietf-netconf-udp-pub-channel-05, 11 March 2019, <http://www.ietf.org/internet-drafts/draft-ietf-netconf-udp-pub-channel-05.txt>.

[I-D.voit-netmod-yang-notifications2]
Voit, E., Bierman, A., Clemm, A., and T. Jenkins, "YANG Notification Headers and Bundles", Work in Progress, Internet-Draft, draft-voit-netmod-yang-notifications2-00, 24 February 2017, <http://www.ietf.org/internet-drafts/draft-voit-netmod-yang-notifications2-00.txt>.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, DOI 10.17487/RFC4287, December 2005, <https://www.rfc-editor.org/info/rfc4287>.

[RFC6202] Loreto, S., Saint-Andre, P., Salsano, S., and G. Wilkins, "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP", RFC 6202, DOI 10.17487/RFC6202, April 2011, <https://www.rfc-editor.org/info/rfc6202>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <https://www.rfc-editor.org/info/rfc7232>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/info/rfc7252>.

[RFC8639] Voit, E., Clemm, A., Gonzalez Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Subscription to YANG Notifications", RFC 8639, DOI 10.17487/RFC8639, September 2019, <https://www.rfc-editor.org/info/rfc8639>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/info/rfc8641>.

[RFC8650] Voit, E., Rahman, R., Nilsen-Nygaard, E., Clemm, A., and A. Bierman, "Dynamic Subscription to YANG Events and Datastores over RESTCONF", RFC 8650, DOI 10.17487/RFC8650, November 2019, <https://www.rfc-editor.org/info/rfc8650>.

[sse] WHATWG, "HTML Living Standard -- 9.2 Server-sent events", n.d., <https://html.spec.whatwg.org/multipage/server-sent-events.html#server-sent-events>.

Acknowledgements

The need for a Series Transfer Pattern has been made clear by a number of people that contribute to the IRTF Thing-to-Thing Research Group (T2TRG), e.g. Matthias Kovatsch and Henk Birkholz (both of whom also provided feedback on an early draft). Henk also contributed further examples for the use of Series Transfers in protocols.

Authors' Addresses

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org

Klaus Hartke
Ericsson
Torshamnsgatan 23
16483 Stockholm
Sweden

Email: klaus.hartke@ericsson.com