INTERNET-DRAFT 5 ISSUES are highlighted like this.

<draft-ietf-ipp-notify-get-poll-00.txt>

Carl-Uno Manros
Tom Hastings
Robert Herriot
Xerox Corp.
Harry Lewis
IBM, Corp.

February 2, 2000 ~~December 7, 1999~~

Internet Printing Protocol/1.1: **The 'ipp-notify-pollget' Notification Polling Method**

Status of this Memo

Abstract

The IPP notification specification [ipp-ntfy] is an OPTIONAL extension to IPP/1.0 and IPP/1.1 that requires the definition of one or more delivery methods for dispatching event notification reports to Notification Recipients. This document describes the semantics and syntax of the 'ipp-notify-pollget' event notification delivery method. For this delivery method, the client uses an explicit IPP Get-Notifications Printer operation in order to request (pull) event Notifications from the IPP Printer.

When a Printer supports the 'ipp-notify-poll' delivery method, it queues Notification events for a window of time for each Subscription object. Notification Recipients poll these Subscription objects at the rate specified by the time window. The Get-Notifications request indicates whether the client wants to receive all ~~future~~ pending events Notifications for (1) any Subscription for which the client is the owner or (2) a particular Subscription object. The Get-Notifications operation retrieves all pending Notifications that occurred for an interval of time in the past for ~~one or more specified subscription-ids~~the requested Subscription objects. The Printer returns the all pending Notifications along with two time intervals that specify the next time window: one is the minimum interval that the client should wait before performing another Get-Notifications on the subscription-id and the other is the maximum interval that the Printer is guaranteed to keep any new Notifications associated with the subscription-id.

The Printer may keep the channel open if the minimum interval is sufficiently short, but in any case the client performs a new Get-Notifications operation each time it wants more Notifications. Since the client will be making Get-Notification requests before the time window expires, the Printer will, on occasion, return the same event Notification in two successive responses. The later ones in the previous response will become the earliest in the next response. The client is expected to filter out these duplicates which is easy to do because of the sequence number in each Notification. ~~Each Get-Notifications request returns a new set of Notifications, that is, it does not return any returned in previous responses. In either case, the event Notifications are returned as MIME multi-part-related responses to the Get-Notifications request. The HTTP channel is kept open, so that subsequence event Notifications are returned using additional MIME multi-part-related responses.~~

The full set of IPP documents includes:

Design Goals for an Internet Printing Protocol [RFC2567]
Rationale for the Structure and Model and Protocol for the Internet Printing Protocol [RFC2568]
Internet Printing Protocol/1.1: Model and Semantics [ipp-mod]
Internet Printing Protocol/1.1: Encoding and Transport [ipp-pro]
Internet Printing Protocol/1.1: Implementer's Guide [ipp-iig]
Mapping between LPD and IPP Protocols [RFC2569]
Internet Printing Protocol/1.0 & 1.1: Event Notification Specification [ipp-ntfy]

The "Design Goals for an Internet Printing Protocol" document takes a broad look at distributed printing functionality, and it enumerates real-life scenarios that help to clarify the features that need to be included in a printing protocol for the Internet. It identifies requirements for three types of users: end users, operators, and administrators. It calls out a subset of end user requirements that are satisfied in IPP/1.0. A few OPTIONAL operator operations have been added to IPP/1.1.

The "Rationale for the Structure and Model and Protocol for the Internet Printing Protocol" document describes IPP from a high level view, defines a roadmap for the various documents that form the suite of IPP specification documents, and gives background and rationale for the IETF working group's major decisions.

The "Internet Printing Protocol/1.1: Model and Semantics" document describes a simplified model with abstract objects, their attributes, and their operations that are independent of encoding and transport. It introduces a Printer and a Job object. The Job object optionally supports multiple documents per Job. It also addresses security, internationalization, and directory issues.

The "Internet Printing Protocol/1.1: Encoding and Transport" document is a formal mapping of the abstract operations and attributes defined in the model document onto HTTP/1.1 [RFC2616]. It defines the encoding rules for a new Internet MIME media type called "application/ipp". This document also defines the rules for transporting over HTTP a message body whose Content-Type is "application/ipp". This document defines a new scheme named 'ipp' for identifying IPP printers and jobs.

The "Internet Printing Protocol/1.1: Implementer's Guide" document gives insight and advice to implementers of IPP clients and IPP objects. It is intended to help them understand IPP/1.1 and some of the considerations that may assist them in the design of their client and/or IPP object implementations. For example, a typical order of processing requests is given, including error checking. Motivation for some of the specification decisions is also included.

The "Mapping between LPD and IPP Protocols" document gives some advice to implementers of gateways between IPP and LPD (Line Printer Daemon) implementations.

The "Event Notification Specification" document defines OPTIONAL operations that allow a client to subscribe to printing related events. Subscriptions include "Per-Job subscriptions" and "Per-Printer subscriptions". Subscriptions are modeled as Subscription objects. Four other operations are defined for subscription objects: get attributes, get subscriptions, renew a subscription, and cancel a subscription.

Table of Contents

1 Introduction

IPP printers that support the OPTIONAL IPP notification extension [ipp-ntfy] either a) accept, store, and use notification subscriptions to generate event Notification reports and implement one or more delivery methods for notifying interested parties, or b) support a subset of these tasks and farm out the remaining tasks to a Notification Delivery Service. The 'ipp-getipp-notify-poll' event notification delivery method specified in this document defines a Get-Notifications operation that may be used in a variety of notification scenarios. Its primary intended use is for clients that want to be Notification Recipients. to explicitly request (pull) event Notifications from the IPP Printer upon request. However, the Get-Notifications operation may also be used by Notification Delivery Services to request (pull) event Notifications from an IPP Printer for subsequent distribution to the Ultimate Notification Recipients. The HTTP channel is kept open, so that subsequence event Notifications are returned using additional MIME multi-part-related responses.

When a Printer supports the 'ipp-notify-poll' delivery method, it queues Notification events for a window of time for each Subscription object. Notification Recipients poll these Subscription objects at the rate specified by the time window. The Get-Notifications request indicates whether the client wants to receive all pending events Notifications for (1) any Subscription for which the client is the owner or (2) a particular Subscription object. The Get-Notifications operation retrieves all pending Notifications that occurred for an interval of time in the past for for one or more specified subscription idsthe requested Subscription objects. The Printer returns all pending Notifications along with two time intervals that specify the next time window: one is the minimum interval that the client should wait before performing a Get-Notifications on the subscription-id and the other is the maximum interval that the Printer is guaranteed to keep any new Notifications associated with the subscription-id.

The Printer may keep the channel open if the minimum interval is sufficiently short, but in any case the client performs a new Get-Notifications operation each time it wants more Notifications. Since the client will be making Get-Notification requests before the time window expires, the Printer will, on occasion, return the same event Notification in two successive responses. The later ones in the previous response will become the earliest in the next response. The client is expected to filter out these duplicates which is easy to do because of the sequence number in each Notification. The reason for not removing the Notifications from the Subscription object with every Get-Notifications request, is so that multiple Notification Recipients can be polling the same subscription object. This is useful if you are logged in to several desktops at the same time and want to see the same events at both places.Each Get-Notifications request returns a new set of notifications, that is, it does not return any returned in previous responses.

2 Terminology

This section defines the following additional terms that are used throughout this document:

REQUIRED: if an implementation supports the extensions described in this document, it MUST support a REQUIRED feature.

OPTIONAL: if an implementation supports the extensions described in this document, it MAY support an OPTIONAL feature.
Notification Recipient - See [ipp-ntfy]
Subscription object - See [ipp-ntfy]
Ultimate Notification Recipient - See [ipp-ntfy]

3 Model and Operation

In the IPP Notification Model [ipp-ntfy], one or more Per-Job Subscriptions can be supplied in the Job Creation operation or OPTIONALLY as subsequent Create-Job-Subscription operations; one Per-Printer Subscription can be supplied in the Create-Printer operation. The client that creates these Subscription objects becomes the owner of the Subscription object.

When creating each Subscription object, the client supplies the "notify-recipient" (uri) attribute. The "notify-recipient" attribute specifies both a single Notification Recipient that is to receive the Notifications when subsequent events occur and the method for Notification delivery that the IPP Printer is to use. For the 'ipp-notify-poll' Notification delivery method defined in this document, there is no notify-recipient because the Printer waits for one or more~~some~~ clients to ask for Notifications from a Subscription object rather than sending them. ~~the notification method is 'ipp-get', and the Notification Recipient is omitted, since~~ Rather, any client that is authenticated (1) as an operator or administrator or (2) as the owner of the Subscription object can initiate a Get-Notifications operation for that Subscription object. Therefore, any Printer that supports the 'ipp-notify-poll' notification delivery method MUST queue event Notifications for a sliding window of time for each Subscription object. Thus a single user can login at different places, say his/her office, the lab, and/or several desktops in the same room, and receive the same event Notifications from a single Subscription object.

~~For the 'ipp-get' notification delivery method defined in the document, the client who created the Subscription objects is also the Notification Recipient.~~ The client issues a Get-Notifications Printer operation in order to initiate the delivery of the ~~next event~~ pending Notifications ~~that occur~~held by the Printer for the ~~specified subscription id~~Subscription objects requested. The client can indicate in the Get-Notifications request whether it wants to receive all ~~future event~~pending Notifications for (1) any existing ~~or future~~ Subscription objects for which it is the owner or (2) ~~a~~ particular Subscription object(s) (for which it MUST be the owner or have read-access rights). In either case, the Notifications are returned ~~as MIME multi-part-related~~in a response~~s~~ to the Get-Notifications request.

If the client requests a persistent channel and if the Printer has returned minimum intervals that are sufficiently short, then the Printer keeps the channel open. ~~The HTTP channel is kept open for an indefinite period, so that the IPP Printer continues to return additional parts of the MIME multi-part-related responses for each event Notification as it occurs.~~ Either the client or the IPP Printer can disconnect the HTTP connection. ~~However, if the IPP Printer grants an HTTP connection it SHOULD disconnect only under unusual circumstances.~~

ISSUE 01: Should it be possible for a client to ask for the Per-Job Subscriptions for a particular job using a "job-id", instead of the subscription-id, which currently isn't returned by a Job Creation operation?

~~ISSUE 01: Is there a limit to the number of outstanding Get-Notifications requests that an IPP Printer supports? What is this number? How does it relate to the maximum number of Subscriptions? Can the client determine the number?~~

~~ISSUE 02: Should an implementation be able to queue event Notifications, so that a client can get event Notifications that had occurred prior to the Get-Notifications? If so, how long does the IPP Printer keep the event Notifications before discarding them (for this delivery method only)? The lease time of the Subscription object? If this is possible, should the subscriber get to say whether to queue or not, or is it just baked into the implementation. If the former, does the subscriber indicate via a parameter in the notification method URL? If the latter, how does a client discover whether event Notifications are queued or not? Should we have two different notification methods, one the queues and one that doesn't?~~

4 Get-Notifications operation

This REQUIRED operation allows the client to request that ~~future event~~pending Notifications be delivered as ~~MIME multi-part-related~~ a response~~s~~ to this request. The client MUST be the owner or have write-access rights of the Subscription objects that are involved and the delivery method specified when the Subscription objects were created MUST be ~~'ipp-get~~ipp-notify-poll~~'unspecified~~. When the Printer creates a Subscription Object, either with a ~~job-creation~~Job Creation operation or with a Create-Printer-Subscription or Create-Job-Subscription operation and a subscription object contains ~~a NULL~~the 'ipp-notify-poll' value for the “notify-recipient” operation attribute, the Printer returns a minimum and maximum interval in the response. The client SHOULD perform a Get-Notifications operation after the minimum interval and if the Printer receives the Get-Notifications before the maximum interval has elapsed, it MUST have all of the Notifications that has occurred since the Subscription object was created. ~~However, the client can and SHOULD issue the Get-Notifications request before having created any Subscription objects, in order not to miss any event Notifications.~~

ISSUE 02: Is there anything useful that we could define for the rest of the "notification-recipient" (uri) attribute, since there is no recipient address needed after the 'ipp-notify-poll://' since the recipient(s) poll?

The IPP Printer MUST accept the request in any state (see [ipp-mod] "printer-state" and "printer-state-reasons" attributes) and MUST remain in the same state with the same "printer-state-reasons".

~~Current "printer-state"~~	~~New "printer-state"~~	~~new "printer-state-reasons"~~	~~IPP Printer's response status code and action:~~
~~'idle'~~	~~'idle'~~	~~no change~~	~~'successful-ok'~~
~~'processing'~~	~~'processing'~~	~~no change~~	~~'successful-ok'~~
~~'stopped'~~	~~'stopped'~~	~~no change~~	~~'successful-ok'~~

~~ISSUE 03: What "printer-state-reasons" might cause an error return, if any? 'paused', 'shutdown', 'quiescent'?~~

Access Rights: The authenticated user (see [ipp-mod] section 8.3) performing this operation must either be the Subscription object owner (as determined when the Subscription object was created by the Job Creation operation, Create-Job-Subscription, or Create-Printer-Subscription operations) or an operator or administrator of the Printer object (see [ipp-mod] Sections 1 and 8.5). Otherwise, the IPP object MUST

reject the operation and return: 'client-error-forbidden', 'client-error-not-authenticated', or 'client-error-not-authorized' as appropriate.

4.1 Get-Notifications Request

The following groups of attributes are part of the Get-Notifications Request:

Group 1: Operation Attributes

Natural Language and Character Set:
The "attributes-charset" and "attributes-natural-language" attributes as described in [ipp-mod] section 3.1.4.1.

Target:
The "printer-uri" (uri) operation attribute which is the target for this operation as described in [ipp-mod] section 3.1.5.

Requesting User Name:
The "requesting-user-name" (name(MAX)) attribute SHOULD be supplied by the client as described in [ipp-mod] section 8.3.

"subscription-ids" (1setOf integer(1:MAX)):
The client OPTIONALLY supplies this attribute. The Printer object MUST support this attribute. It is an integer value that identifies one or more ~~the~~ Subscription objects for which event Notifications are being requested. If the client supplies this attribute, but none of the Subscription objects ~~is~~ are ~~not~~ found, the IPP Printer MUST return the 'client-error-not-found' status code. If some are found and others are not, the ones that are not found are return in the Unsupported Attributes.

If the client does not supply this attribute, then the IPP Printer returns event Notifications for all Subscription objects for which the client is the owner and the "notify-recipients" attribute is ~~NULL~~'~~ipp-get~~'ipp-notify-poll'. It is not an error if there are currently no Subscription objects for this client; the response then contains no Notifications.~~the client can create Subscription objects later that will start returning event Notifications as responses to this operation~~.

~~1.2~~4.2 Get-Notifications Response

The Printer object returns either an immediate error response or a successful response with status code: 'successful-ok' when the first event occurs, i.e., when the Printer delivers the first event Notification.

Group 1: Operation Attributes

Status Message:
In addition to the REQUIRED status code returned in every response, the response OPTIONALLY includes a "status-message" (text(255)) and/or a "detailed-status-message" (text(MAX)) operation attribute as described in [ipp-mod] sections 13 and 3.1.6.

Natural Language and Character Set:

The "attributes-charset" and "attributes-natural-language" attributes as described in [ipp-mod] section 3.1.4.2.

"minimum-time-interval" (integer(0:MAX)):

The value of this attribute is the minimum number of seconds that SHOULD elapse before the client performs this operation again for these subscription-ids. A client MAY perform this operation at any time, and a Printer MUST respond with all pending Notifications. A client observes this value in order to be a “good network citizen”.

"maximum-time-interval" (integer(0:MAX)):

The value of this attribute is the maximum number of seconds that SHOULD elapse before this client SHOULD issue ~~a Printer receives~~ this operation again for these subscription-ids. A Printer MUST preserve all Notifications that occur for the number of seconds specified by this attribute starting at the time it is sent in a response. A client MAY perform this operation at any time, and a Printer MUST respond with all pending Notifications. If a Printer receives this operation after this time interval, it ~~may~~ SHOULD have discarded some Notifications since the last response.

ISSUE 04 - Or MUST the Printer discard events that occurred earlier than the sliding time window specified by the difference between these two values? Otherwise, the clients may get back a lot of duplicate events on subsequent requests.

Group 2: Unsupported Attributes

See [ipp-mod] section 3.1.7 for details on returning Unsupported Attributes.

If the “~~requested-attributes~~subscription-ids” attribute contained~~s~~ subscription-ids that do not exist, the Printer returns them ~~they are returned~~ in this group as value of the “subscription-ids~~requested-attributes~~” attribute.

Group 3 through N: ~~Generic Object Attributes~~Notification Attributes

The Printer object responds with one event Notification per Group for each pending Notification that meets the criteria specified by the subscription-ids attribute and requesting user name.(see [ipp-ntfy]). ~~If there are multiple events that occur at the same time, the Printer object returns them in separate MIME multi-part-related responses, each as separate IPP operation responses, as well. The HTTP channel is kept open for an indefinite period, so that the IPP Printer continues to return additional parts of the MIME multi-part-related responses for each event Notification as it occurs. ISSUE 04 - Is this correct for MIME multi-part-related responses? This need prototyping.~~

5 Extension to Print-Job, Print-URI, Create-Job, Create-Printer-Subscription and Create-Printer-Subscription

5.1 Response

When Print-Job, Print-URI or Create-Job contains a “job-notify” attribute and the “notify-recipient” is 'ipp-notify-poll'~~NULL~~, the response contains two additional Operation Attributes that pertain to subscriptions.

When Create-Job-Subscription or Create-Printer-Subscription operation contains a “notify-recipient” that is 'ipp-notify-poll'~~NULL~~, the response contains two additional Operation Attributes that pertain to subscriptions.

Group 1: Operation Attributes

"minimum-time-interval" (integer(0:MAX)):
The value of this attribute is the minimum number of seconds that SHOULD elapse before the client performs the Get-Notification operation for the first time with any subscription-ids returned with this job. A client MAY perform the Get-Notification operation at any time, and a Printer MUST respond with all pending Notifications. A client observes this value in order to be a “good network citizen”.

ISSUE 05: if we don’t want to have Job Creation operations return subscription id’s, then allow a "job-ids" operation attribute in the Get-Notifications request in addition to the "subscription-ids" operation attribute.

"maximum-time-interval" (integer(0:MAX)):
The value of this attribute is the maximum number of seconds that SHOULD elapse before a Printer receives the Get-Notification operation for the first time with any subscription-ids returned with this job. A Printer MUST preserve all Notifications that occur for the number of seconds specified by this attribute starting at the time it is sent in a response. A client MAY perform the Get-Notification operation at any time, and a Printer MUST respond with all pending Notifications. If a Printer receives a Get-Notification operation after this time interval, it may have discarded some Notifications since the last response.

6 Encoding

The operation-id assigned for the Get-Notification operation is:

0x00??

and should be added to the next version of [ipp-mod] section 4.4.15 "operations-supported".

This notification delivery method uses the IPP transport and encoding [ipp-pro] for the Get-Notifications operation with one extension:

Instead of defining a new object attribute tag, a Generic Object attributes tag is defined that is used for all new objects, such as Subscription objects, etc. Then this one new tag can also be used for the Get-Notifications response Group 3 tag and subsequent groups in section 4.2:

~~generic-object~~notification-attributes-tag = %x07~~??~~ ; tag of 7~~?~~

7 IANA Considerations

IANA will be asked to register this '~~ipp-get~~ipp-notify-poll' notification delivery scheme.

8 Internationalization Considerations

With the '~~ipp-get~~ipp-notify-poll' method defined in this document, the client cannot request the Human Consumable form by supplying the "notify-text-format" operation attribute (see [ipp-ntfy]). Therefore, the IPP Printer does not have to perform any localization with this notification delivery method. However, the client when it receives the Get-Notifications response is expected to localize the attributes that have the 'keyword' attribute syntax according to the charset and natural language requested in the Get-Notifications request.

9 Security Considerations

The IPP Model and Semantics document [ipp-mod] discusses high level security requirements (Client Authentication, Server Authentication and Operation Privacy). Client Authentication is the mechanism by which the client proves its identity to the server in a secure manner. Server Authentication is the mechanism by which the server proves its identity to the client in a secure manner. Operation Privacy is defined as a mechanism for protecting operations from eavesdropping.

Unlike other event Notification delivery methods in which the IPP Printer initiates the event Notification, with the method defined in this document, the Notification Recipient is the client who issues the Get-Notifications operation. Therefore, there is no chance of "spam" notifications with this method. Furthermore, such a client can close down the HTTP channel at any time, and so can avoid future unwanted event Notifications at any time.

10 References

[ipp-mod]
R. deBry, T. Hastings, R. Herriot, S. Isaacson, P. Powell, "Internet Printing Protocol/1.0: Model and Semantics", <draft-ietf-ipp-model-v11-04.txt>, June, 1999.

[ipp-ntfy]
Isaacson, S., Martin, J., deBry, R., Hastings, T., Shepherd, M., Bergman, R., "Internet Printing Protocol/1.1: IPP Event Notification Specification", <draft-ietf-ipp-not-spec-01.txt>, October 14, 1999.

[ipp-pro]
Herriot, R., Butler, S., Moore, P., Tuner, R., "Internet Printing Protocol/1.1: Encoding and Transport", draft-ietf-ipp-protocol-v11-03.txt, June, 1999.

[rfc2026]
S. Bradner, "The Internet Standards Process -- Revision 3", RFC 2026, October 1996.

[RFC2616]
R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol - HTTP/1.1", RFC 2616, June 1999.

11 Author's Addresses

Robert Herriot
3400 Hillview Ave., Bldg #1
Palo Alto, CA 94304

Phone: 650-813-7696
Fax: 650-813-6860
Email: rherriot@pahv.xerox.com

~~Carl-Uno Manros~~
~~Xerox Corporation~~
~~737 Hawaii St. ESAE 231~~
~~El Segundo, CA 90245~~

~~Phone: 310-333-8273~~
~~Fax: 310-333-5514~~
~~e-mail: cmanros@cp10.es.xerox.com~~

~~Tom Hastings~~
~~Xerox Corporation~~
~~737 Hawaii St. ESAE 231~~
~~El Segundo, CA 90245~~

~~Phone: 310-333-6413~~
~~Fax: 310-333-5514~~
~~e-mail: hastings@cp10.es.xerox.com~~

12 Full Copyright Statement