Tsert::OS Reminder©® Subsystem a Uniform, Internet-Enabled, & Fragment-Based Notification System

Pierre Innocent, Member IEEE
Tsert.Com
contact@tsert.com
http://www.tsert.com/

Abstract – Notification is an old concept in operating system design. Notification systems used to be implemented as system logs, which other tools would then use to generate reports. Some tools send email to the system manager when certain conditions arise.

Our approach is to systematically categorize and manage every type of event that an operating system or a user's interaction, with said operating system, can generate, with the use of visual notifications. Such notifications will, heretofore, be referred to as reminders

The Tsert::OS Reminder©® subsystem provides a way to visually notify the user of events triggered on their desktop; it also provides a simple way for users to exchange messages with each other. Such reminders, akin to visual texting or email, (texting was in Unix systems with commands such as, mesg, and write), can be exchanged across the Internet.

Reminders are deposited on the drive, and picked up by the reminder subsystem; which subsequently displays them to the user, if they are local reminders; otherwise, they are transmitted to the peer desktop.

The transmission takes place with the use of HTTP or XML fragments.

Index Terms – Fragment, Salt, Protocol, Reminder, Notification, Multi-Cast, Salt Value, IP, XML, HTTP.

I. Introduction

The use of notifications is as old as the Unix system. Notifications have been mostly used, to warn system administrators of probable, or possible problems with their installation. These notifications usually took the form of crafted emails to the system administrator, who then had to examine the system logs, in order to fix the problem.

A. Old Approach

The old approach, usually, addressed events related to the operating system and peripherals; and less events related to the user's interaction with the system. And logs, because of the lack of a uniform logging methodology, could not easily be used to track problems in a uniform manner.

Except for the kernel logging facilities, each application found their own way to track reminder type pieces of information. But, unfortunately, even the system logging (syslog) facilities are not used in a uniform manner and do not provide an easy way to display the logged information visually.

II. Solution

Our approach uses visual and text-based reminders. A reminder is a set of entries clearly identifying its type, its source or sender, its creation time, the cause which triggered its creation (subject for email reminders, and purpose for document reminders), its destination, an optional icon name, a display attribute entry, an optional command or request entry, an optional status entry for the command or request, its urgency level, an optional **url** pointing to the full content of the information which triggered the notification, and a text body.

Applications **may** use either the **cause**, **purpose**, or **subject** entry in reminders of any type. The **command** or **request** entry, as well as, the **status** entry are optional.

Every application or daemon, must use the same API to communicate with the reminder subsystem. The API uses **XML/HTML** fragments or **HTTP POST** submissions, and transmits the reminders through **TCP/UDP** connections to peer subsystems. **HTTP GET** and **PUT** may be used for transmission of fragments on **UDP** connections.

A. Fragments

Fragments are sections of text that start and end with an **SGML**-like tag. **SGML** is the precursor language of **HTML**, **XML**, and other derivatives. Fragments are seen as just another **Content-Type** (**fgr/html, fgr/xml, fgr/cfg, fgr/xml+rmr**, **fgr/html+rmr**, etc.); and obey the same cache semantics as other content types.

An **XML/HTML** reminder fragment may contain, a **SALT** load[1]. Fragments may be encrypted with the peer''s or user's public key. They may be prepended with a set of **HTTP** headers for transmission.

When a peer receives a **UDP** fragment, it must first examine whether it is just an **XML/HTML** fragment, or an **HTTP PUT** request, specifying a **content type** of reminder fragment.

Fragments sent on **TCP** connections must always use the **HTTP** protocol, and the fragment or www-url-encoded **content type**.

HTML fragments are delimited by the **OBJECT** tag; and has style attributes, such as **display** and **trigger**. The object is of type **reminder**, and can be instantiated on your desktop as an applet.

III. Reminders

Reminders, as described above, are pieces of text data, used to notify users of events related to their desktop or installation. There are several types of reminders used in our operating system; they are as follows:

1. the Application reminder.
2. the Email reminder.
3. the Task reminder.
4. the Document reminder.
5. the Search reminder.
6. the Device reminder.
7. the Network reminder.
8. The Synchronisation reminder.
9. User-Defined reminder.

The urgency level of reminders is as follows:

1. Information
2. Urgency
3. Warning
4. Criticality

[1]See the Salt protocol white paper

A. Application Reminders

Application reminders are used, by applications installed on your system, to convey information, that they cannot otherwise convey with other reminders. The information **must** be of the type that is specific to the application itself; for example, a database engine may relate some information to the user, about a problem, that is not document or query related.

B. Email Reminders

Email reminders are short pieces of text sent to a peer, instead of a full-blown email. The text of the reminder can be scanned and the information stored by calendaring and PIM type applications. The reminder is seen as **visual texting**, used to inform a given user, about something to do, to remember, or to remember to do, etc..

The XML schema of an email reminder fragment is the same as that of a basic reminder; except for the **cause** entry, which becomes the **subject** entry in email reminders, and the **purpose** entry in document reminders.

C. Task Reminders

A task is a process that is instantiated on a given schedule,. In most operating systems, they usually are shell scripts that are written do to miscellaneous chores on your installation. Task reminders must always indicate the status of the completion of their task; and optionally their instantiation time. Email clients that can be scheduled to retrieve emails, are seen as such tasks.

D. Document reminders

Document reminders are used to track the status of certain documents that are accessed in collaborative environments. They are used to track information such as creation, opening, modification, and closure time.

A user may want to know, when a given **restricted** document is being accessed or modified.

E. Search Reminders

Search reminders are used to tell the user when certain queries, scheduled or not, are successful or not. An information retriever task may be instantiated at regular intervals to retrieve data on the internet; and inform you when certain data have been retrieved.

An email client acting as a task may use the search type of reminder; but **must** clearly indicate itself, a **task**, as the source of the reminder.

F. Device Reminders

Device reminders are used by applications or the kernel to indicate when certain devices are being accessed. Devices, as file system entries, can be created, mounted, and accessed for reading or writing. Kernel or applications **may** choose to indicate when certain devices are being accessed; or use the display attribute of **none,** to indicate that the reminder should simply be logged.

G. Networking Reminders

Network or networking reminders are used to inform the user about events happening with their installation's network stack. Information about any event related to the network stack, such as firewall events, network intrusion, off-line nodes in your Local Area Network (LAN), may be conveyed to the user through network reminders.

H. Synchronisation Reminders

Synchronisation reminders are specific to our operating system. They are used to inform the user when a given host has properly completed a network handshake, with their installation, using the **SALT** protocol. They are also are used to tell the user, about peers not providing the proper **SALT** setting to establish a connection.

Synchronisation reminders may be also be typed as network reminders.

I. Urgency level

The four urgency levels are information, warning, urgency, and criticality. An information notification is related to information which does not have to be accessed or referenced immediately, such as normal email, system and device events, etc..

A warning notification has essentially the same urgency as information notifications; except that they reference information, about **non-critical failures** of commands or requests, which may be of use to a user or system administrator.

An urgent notification is required to be moved to the front of the display queue. Such notifications **must** be about information that a user or system administrator

must see without delay, regardless of source, cause, purpose, or subject.

Critical notifications are related to failures of some sort, such as system, device, or network failures, which need **immediate** attention. They also **must** be moved to the front of the display queue. Such notifications **should** be tracked in a database for a trace of events emanating from their particular source.

IV. Reminder subsystem

The purpose of a reminder subsystem is to centralize the point of access and delivery of reminders. It is to mainly ensure the proper display of reminders; and their storage in user-specified databases.

Only **critical** notifications can override relevancy triggers (next section), and the display attribute of **none**; and they can only be restricted by the subsystem itself, if they become incessant.

A. Relevancy

Relevancy is the criteria which must be satisfied by the reminder subsystem. It must be able to deliver and display only notifications that are relevant to the user.

The user is **required** to specify the types of events, for which notifications are to be displayed. The user must indicate the type of events, their source, the request or command which triggered it, and optionally some keywords related to subject, purpose, cause, or content of the text body.

A simple database of these notification relevancy triggers is kept for each user. The design and development of such databases is left to the implementer's choice.

B. Overload

To prevent the user from being overloaded by needless notifications, despite the use of relevancy triggers; the subsystem schedules the display of notifications on thirty second intervals. Each notification **must** be displayed for no more than 12 seconds; and no less than 5 seconds.

When the subsystem detects that a given source is sending spurious, frequent, or repetitive notifications; it can disallow the reception of any more notifications from said source. The user is **a-priori** notified and asked to confirm such firewall-like action.

V. Conclusion

Our approach gives applications developers, a method to store and manage system and user notifications in a uniform manner. These notifications or reminders, stored in a database or as search indices, can subsequently be easily accessed and referenced for information about the prior or current state of a given installation.

Also, our approach allows users to exchange information that can be tracked through PIM type applications; the same way email information may be scanned and tracked.

VI. Appendix A

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE tsert:reminder [
<!ENTITY copyright "Copyright 2008-2009, Tsert.com, All Rights Reserved.">

<!ELEMENT reminder:notification (
      ( reminder:source | reminder:sender )
      ( reminder:cause | reminder:purpose | reminder:subject ) ?
      ( ( reminder:request | reminder:command )  ?
      reminder:destination
      reminder:creation-time
      reminder:image ?
      reminder:resource ?
      reminder:text
)>

<!ELEMENT reminder:source (#PCDATA)>
<!ELEMENT reminder:sender (#PCDATA)>
<!ELEMENT reminder:cause (#PCDATA)>
<!ELEMENT reminder:purpose (#PCDATA)>
<!ELEMENT reminder:subject (#PCDATA)>
<!ELEMENT reminder:request (#PCDATA)>
<!ELEMENT reminder:command (#PCDATA)>
<!ELEMENT reminder:destination (#PCDATA)>
<!ELEMENT reminder:creation-time (#PCDATA)>
<!ELEMENT reminder:image EMPTY>
<!ELEMENT reminder:resource EMPTY>
<!ELEMENT reminder:text (#PCDATA)>

<!ATTLIST reminder:notification type ( task|app|search|device|network|email|synchro|doc|CDATA )>
<!ATTLIST reminder:notification urgency ( info|warning|urgent|critical ) “info”>
<!ATTLIST reminder:notification display ( none|fade-in|zoom-in|slide|scroll|static ) “static”>

<!ATTLIST reminder:request status ( success|failure|cancelled|completed )>
<!ATTLIST reminder:command status ( success|failure|cancelled|completed )>
<!ATTLIST reminder:resource url CDATA #REQUIRED>
<!ATTLIST reminder:image src CDATA #REQUIRED>

]>
```

VII. References

[1] Chi-Huang Chiu, Ruey-Shyang Wu, Chi-Io Tut, Hsien-Tang Lin, Shyan-Ming Yuan Nat, "Next Generation Notification System Integrating Instant Messengers and Web Service", IEEE Explore, 1781-1786, November 2007.

[2] Jan Willem Streefkerk. Myra P. van Esch-Bussemakers, Mark A. Neerincx, "Field evaluation of a mobile location-based notification system for police officers", ACM Portal, 101-108, 2008

[3] Mohamed, Nader Al-Jaroodi, Jameela Jawhar, Imad , "A generic notification system for Internet information", IEEE-Explore, 166-171, July 2008

[4] Innocent, Pierre, "Salt Protocol an Identity-Based Authentication Protocol using Synchronized Systems", white-paper, www.tsert.com, November 2008.