INTERNET-DRAFT Mark Day
Expires: September 13, 1998 Lotus Development Corporation

Simple General Awareness Protocol (SGAP) Revision 1

draft-day-sgap-01.txt

1. Status of this Memo

2. Abstract

The Simple General Awareness Protocol (SGAP) provides notifications of changes to small data items. The changes are selectively made available to a large collection of viewers. This facility is most useful for a class of applications variously called people locators, colleague-awareness tools, people browsers, or "buddy lists".

3. Contents

4. Introduction

Synchronous collaboration in a network environment is determined by the characteristics of people and how they work together. We have found it useful to divide synchronous collaboration into three phases: awareness, conversation, and sharing. A collaboration proceeds through the phases in that order. Multiple overlapping and returning sequences through this model are possible

i. Awareness: A user first becomes aware that other users, groups, or resources of interest are available. Awareness is comparable to the function of peripheral vision.

ii. Conversation: Having become aware of those others, the user starts one or more conversations or conferences with those people (via various media, such as voice, text, or video).

iii. Sharing: Being engaged in conversations, the user finds it useful to share documents or other objects. Such sharing adds value in two ways. First, it allows for richer collaboration than is possible with conversation alone, such as jointly editing a document or playing a real-time game. Second, it allows for conversational ellipsis: for example, a gesture at a sentence accompanied by the phrase "I don't like this" instead of "I don't like the sentence halfway down page 3 that starts with 'Synchronous.'"

The Simple General Awareness Protocol (SGAP) is intended to be used primarily for awareness, the first phase of the process. SGAP is not intended as a mechanism for conversation or sharing; there are a number of technologies that can be used for various forms of conversation and sharing, and relatively few that allow for the features required of an awareness system.

In addition, the requirements in the two contexts are fundamentally different. An awareness system involves the selective exposure of small amounts of information to large numbers of others, with relatively loose consraints on latency. A conversation and sharing system involves the sharing of large amounts of information (high-bandwidth channels) among a relatively small group, with relatively tight constraints on latency. To use a telephone-system analogy, we consider awareness to be largely "signalling" for the higher-bandwidth "connection" represented by the conversations and sharing.

5. Protocol Model

Conceptually, a server implements a matrix, with a row for each distinct item to be shared and a column for each distinct viewer with whom that information should be shared. The intersection of a column and row is called a cell, and corresponds to the view of an item delivered to a viewer.

5.1 Items

Each item has a distinct item name. The item name is used simply as an identifier for the item; in general, there are no additional semantics (however, a small number of item names are significant to the server). For example, "www.lotus.com" is different from "http://www.lotus.com", though both are legal item names, as is any UTF-8 encoded string of non-zero length (such as "Bill Clinton" or "X%66 52 *$#").

5.2 Viewers

Each viewer also has a distinct viewer name. Again, the viewer name is used simply as an identifier for the viewer; in general, there are no additional semantics (again, a small number of viewer names are significant to the server).

5.3 Viewer and Item Sharing Name

Although distinct items have distinct names, and distinct viewers have distinct names, it is acceptable for an item and a viewer to have the same name. Indeed, such name-sharing is the typical case.

5.4 Authentication

The protocol assumes an authenticated connection between client and server. The authenticated client entity is referred to as the principal.

5.5 Properties

A property is the smallest unit created, destroyed, or modified by the protocol. A property consists of a name, a type, and a value. Both the name and type of a property are UTF-8 encoded text strings, with relatively few restrictions on the form of the strings. A property's value is zero or more bytes of arbitrary data. Architecturally, a value may be up to 4 gigabytes, but servers may set a smaller bound on the size of values. A server MUST NOT break the connection only because a client sends an excessively-large value. Instead, the server must send an error message, then count bytes past the offending element and continue with the next usable request.

5.6 Declarations

A client declares its use of a given name. Usually, such a declaration means that the client both acts as the item with that name, and acts as the viewer with that name; however, a client MAY decline one of the roles when it declares a name. The server MAY refuse to allow a given principal to declare particular names. The server MAY also limit the number of names that a given principal may declare. Finally, the server MAY require that a given principal use one of a restricted set of names.

A client declared as a given item can affect (change the properties of) the row corresponding to that item. A client can only affect a row if it has declared itself as that item. A client declared as a given viewer can see (get the values of) the column corresponding to that viewer. A client can only see a column if it has declared itself as that viewer.

5.7 Notifications

A client may enable or disable notifications on any item that it can see. When notifications are enabled on an item, any change in the properties of that item cause a notification message to be sent to that client.

5.8 Default Viewer

There is a special viewer column representing the default viewer. This column does not have an associated viewer name, and accordingly can never be directly accessed by a viewer. However, a client declared as an item can determine which properties are in the default column for that item, as well as determining which viewer names actually map to the default column for that item.

The default-viewer column allows the possibility of both default-accessible and default-inaccessible sharing of information. For a default-accessible item, the item's value appears in the default column's cell. Viewers to be excluded are named explicitly, with no value (or substitute values) appearing in the cells of those columns. For a default-inaccessible item, no value appears in the default column's cell. Viewers who can access the item are named explicitly, with the appropriate value appearing in the cells.

5.9 Multiple Items and Viewers

A single client-to-server transport connection can carry out commands on behalf of an arbitrary number of items and viewers. In the case of a "buddy list," the expected behavior is to act on behalf of a single item and a single viewer, both representing the person using the client.

5.10 Contexts

A single physical server can implement a number of logical servers, each with its own matrix unrelated to the others but (usually) using the same rules for control of names. Each such logical server is distinguished by a different value of the context field. Contexts are named by UTF-8 encoded strings. The server's default context is named by the zero-length string. The server MAY refuse to allow a given principal to use particular context names. The server MAY also limit the number of names that a given principal may declare. Finally, the server MAY require that a given principal use one of a restricted set of names.

6. Server-Server Protocol Not a Special Case

Multiple servers can coordinate their state using the existing protocol. The protocol allows both a "publish" model and a "subscribe" model, favoring neither over the other.

Consider a server P that has some items (x,y,z) being updated by clients. To "publish" those items and make them available to Q's clients, P connects to another server Q and declares the names (x,y,z). From Q's perspective, P is just another client (although P's authentication to Q likely establishes that P's principal has more privileges than the typical client).

Correspondingly, to "subscribe" to those items, a third server R can connect to P and declare the names (x,y,z), choosing to be only in the viewer role on those names.

Although SGAP supports the publication or subscription of items across multiple servers, the protocol does not specify how clients or servers arrange for such publication or subscription to take place. Note that there is no mechanism in the protocol that would either prevent or detect a request loop among servers.

7. Special Server Behavior

Some properties provide information to the server about how to treat an item. For example, some items should be deleted if the reponsible client is disconnected, while others should persist; this difference in behavior is encoded in one or more special properties understood by the server.

The properties understood by a server are collectively called the server schema. Server schemas are defined independently of the protocol. However, the mechanism to determine the schema of a server is fixed by the protocol.

The server schema of a given server is exposed as the item named "SGAP:Schema-Root". Within that item, the properties named "SchemaName" and "SchemaVersionNumber" are guaranteed to be defined. The value of SchemaName is a UTF-8 encoded text string. The value of SchemaVersionNumber is a 4-byte array, representing an unsigned integer in network order.

8. Protocol Specification

8.1 Model

We consider an awareness service to be a map:

context X viewer name X item name -> property*

That is, for each triplet of <context,viewer name,item name>, there is a collection of zero or more properties.

8.2 Summary of Messages

In general, messages are carried via a reliable bytestream transport,

such as TCP. The following table summarizes the protocol messages and
their function.

Message	Meaning
Init	Start interaction with server
Declare	Claim right to act on behalf of viewers and/or items
Create	Create new properties of an item exposed to a viewer
Modify	Modify the properties of an item exposed to a viewer
Delete	Delete the properties of an item exposed to a viewer
Split Viewers	Inform server that these viewers should see a private (non-default) view of the item
Merge Viewers	Inform server that these viewers should see the default view of the item
List Viewers	Get the names of viewers with private views of an item
Viewer List	Reply containing the names of viewers with private views of an item
Fetch	Get the properties provided to a viewer for a set of items
Fetch Reponse	Reply containing the properties for a set of items
Enable	Allow delivery of change notifications
Disable	Suppress delivery of change notifications
Creation	Notify of property creation
Modification	Notify of property modification
Deletion	Notify of property deletion
OK	Acknowledge successful completion
Error	Explain an error condition

8.3 Header Structure

The header of each message consists of a one-byte version number, a
one-byte opcode, one reserved byte, one default-flag byte, and a
4-byte length (in network order) of the following body in bytes, not
including the header or length itself.

```
    +--------------+--------------+--------------+--------------+
    |     0x85     |    opcode    |     0x00     | default-flag |
    +--------------+--------------+--------------+--------------+
    |              length in bytes of following body            |
    +-----------------------------------------------------------+
    |                            body                           |
    +                              .                            +
    |                              .                            |
    +                              .                            +
```

The version number MUST be set to 0x85 in this version of the protocol.

The opcode identifies the operation to be carried out in response to the message.

The reserved byte MUST be set to 0x00 in this version of the protocol and MUST be ignored by implementations.

The default-flag byte is used to indicate whether an operation applies to the default cell(s) and/or all other cell(s). The flag is needed because the default cells have no name: they correspond to every viewer for which the server does not have a viewer-specific cell. In addition, without using such a mechanism, there is no way to name "all viewers other than default."

The following table lists the acceptable values for the default-flag byte and their interpretation.

0x00	All viewer cells to which the operation applies are explicitly named in the body.
0x01	The default viewer cell(s) are affected in addition to any cells explicitly named in the body.
0x02	All cells except the default viewer cell(s) are affected.
0x03	All cells of the item(s) are affected.

No other values are legal. That is, the two low-order bits serve as flags, and the other bits MUST be zero.

The default-flag byte is only used for a few message types, but there would be no advantage to putting its flags into the body of those messages. Neither the version number nor the opcode requires more than a single byte, while it is advantageous for some machines to have the 4-byte length aligned on a 4-byte boundary. So the default-flag byte uses a byte that would otherwise be "wasted" in the header.

8.4 Body Encoding

A "vector" is encoded as a 4-byte unsigned integer in network order, followed by the number of elements determined by that integer's value, followed by 0 to 3 padding bytes. The number of padding bytes is chosen so that the entire vector's length in bytes is a multiple of 4**. Padding bytes are chosen to alternate between the values 0xAC and** 0xDC. Implementations MUST ignore the value of padding bytes received.

A "String" is a vector of bytes. The bytes contain the UTF8 encoding of character data. There is no null character terminating the string.

A "Property" is two Strings (name and type) followed by a vector of bytes. In contrast to the Strings, the value vector may contain arbitrary bytes, including values that are not allowed in a UTF-8 encoding. A Property is considered to be a single element. Thus, a vector of Properties starts with a count of the number of Properties, not the (three times larger) number of component vectors.

8.4.1 Types

The following type names and meanings are defined by the protocol.

Type Name	bytes in value	Interpretation
SGAP:boolean	1	0x00 = false 0x01 = true other values illegal
SGAP:ternary	1	0x00 = no 0x01 = yes 0x02 = maybe
SGAP:byte	1	arbitrary byte value
SGAP:int	4	32-bit 2's complement integer in network order
SGAP:unsigned	4	32-bit unsigned integer in network order
SGAP:string	arbitrary	UTF-8 encoded
SGAP:javastring	2+arbitrary	UTF-8 encoded with 16-bit prefix count
SGAP:cstring	arbitrary	ASCII with null terminator
SGAP:xml-1.0	arbitrary	expression, XML 1.0 syntax
SGAP:MIME	arbitrary	MIME-encoded value

Other type names MAY be defined. Such type names SHOULD take the form of an internet domain name [RFC-1034] identifying the naming entity, followed by a colon ":", then the type name assigned by the naming entity. Thus "lotus.com:DominoInteger", "ietf.org:IANARegistryEntry", and "cl.cam.ac.uk:2@3" are all acceptable type names.

8.5 Messages

For each message, we present information in the following form (using the Split Viewers message as an example):

```
     8.5.5 Split Viewers

     Opcode: 0x05

     ContextName: String
     ItemName: String
     ViewerNames: vector of String

     Indicates to the server that the named viewers are not to be
```

given the default viewer cell for the named item in the named context. The default-flag byte must be cleared (value of 0x00).

The numbered section heading gives the name of the message (Split Viewers, in this case).

The next line gives the value for the opcode byte, in hexadecimal. Accordingly, a message with the opcode 5 is a Split Viewers message.

The next lines give the encoding of the message's body. The body of a Split Viewers consists of three elements in an implicit sequence: a String, another String, and a vector of Strings. The names of the elements serve as a clue to their function, and provide a way for the explanatory text to refer to these elements. The names of elements (such as "ContextName") are for explanation only and are not explicitly represented in protocol messages.

The final paragraph summarizes the function of the message and the meaning of its elements, including any constraints on the values in the body or in the header.

8.5.1 Init

Opcode: 0x01

This message MUST be sent to the server before any other SGAP messages are sent. It provides the server an opportunity to initialize state related to the authentication of the transport connection and its effect on the server's behavior for this client.

8.5.2 Declare

Opcode: 0x02

```
ContextName: String
Name: String
MultiNames: vector of NameDeclaration

     where NameDeclaration is

     DeclaredName: String
     NameModifiers: vector of integer
```

Declares to the server that the client will use Name as its name. The client can affect the item with that name and act as the viewer with that name.

Name and MultiNames are mutually exclusive: if Name is a non-zero-length String, MultiNames MUST be a zero-length vector. If

Name is a zero-length String, MultiNames MUST NOT be a zero-length vector.

For each NameDeclaration in MultiNames, the DeclaredName MUST NOT be a zero-length String. Implementations MUST reject requests in which the same DeclaredName appears more than once.

Elements of NameModifiers can be any of the following:

0x1 "ItemOnly"	the client may affect the named item but will not act as the named viewer.
0x2 "ViewerOnly"	the client may act as the named viewer but will not affect the named item.
0x3 "Exclusive"	the client insists that it is the only declarer of this name in this form.
0x4 "ItemViewer"	the client may affect the named item and may act as the named viewer.

An implementation MUST reject any request that includes more than one of "ItemOnly", "ViewerOnly", or "ItemViewer" in NameModifiers.

"Exclusive" alone or with "ItemViewer" means that no other client may affect the item by that name or act as the viewer by that name. The item is still potentially available to other viewers.

"Exclusive" and "ItemOnly" together mean that no other client may affect the item by that name, although other clients may declare the same name "ViewerOnly".

"Exclusive" and "ViewerOnly" together mean that no other client may act as the viewer with that name, although other clients may declare the same name "ItemOnly".

8.5.3 Create

Opcode: 0x03

ContextName: String
ItemName: String
ViewerNames: vector of String
NewProperties: vector of name-value pairs

Creates new properties contained within the named item in the named context. Affects the cells determined by the combination of the default-flag byte and ViewerNames.

A single create request takes place atomically. No client sees the result of a partially executed create.

An implementation MUST reject the creation of a property that already exists in any of the cells affected.

Causes Creation notifications (see 8.5.14) to be delivered to viewers that have enabled notifications.

8.5.4 Modify

Opcode: 0x04

ContextName: String
ItemName: String
ViewerNames: vector of String
NewProperties: vector of name-value pairs

Replaces the properties contained within the named item in the named context. Affects the cells determined by the combination of the default-flag byte and ViewerNames.

A single modify request takes place atomically. No client sees the result of a partially executed modify.

An implementation MUST reject the modification of a property that does not exist in any of the cells affected.

Causes Modification notifications (see 8.5.15) to be delivered to viewers that have enabled notifications.

8.5.5 Delete

Opcode: 0x05

ContextName: String
ItemName: String
ViewerNames: vector of String
NewProperties: vector of name-value pairs

Deletes the properties contained within the named item in the named context. Affects the cells determined by the combination of the default-flag byte and ViewerNames.

A single delete request takes place atomically. No client sees the result of a partially executed delete.

An implementation MUST reject the deletion of a property that does not exist in any of the cells affected.

Causes Deletion notifications (8.5.16) to be delivered to viewers that have enabled notifications.

8.5.6 Split Viewers

Opcode: 0x06

ContextName: String
ItemName: String
Copy: Byte
ViewerNames: vector of String

Indicates to the server that the named viewers are not to be given the default viewer cell for the named item in the named context.

If Copy has the value 0x01, then the current properties of the default viewer cell are copied into the new cells so that no change in state is visible to the named viewers; no notifications are sent to any clients that have notifications enabled. Otherwise, the new cells contain no properties, and the affected viewers receive Destroyed notifications for all of the properties that they could see before.

The default-flag byte must be cleared (value of 0x00).

8.5.7 Merge Viewers

Opcode: 0x07

ContextName: String
ItemName: String
ViewerNames: vector of String

Indicates to the server that the named viewers are to be given the default viewer cell for the named item in the named context. The default-flag byte must be cleared (value of 0x00).

For affected clients that have enabled notifications, the following rules are used to determine notifications:

i. For any property that is no longer present in the default cell, the clients receive a Destroyed notification.

ii. For any property present in the default cell that was not present in the original cell, the clients receive a Created notification.

iii. For any property present in both original and default cell, the client receives a Changed notification if and only if the type and/or value of the property differs.

8.5.8 List Viewers

Opcode: 0x08

ContextName: String

ItemName: String

Requests a list of the names of named viewers (those viewers with other than the default view of the named item). The default-flag byte must be cleared (value of 0x00).

The server MUST reject this request if sent by any client that has not succesfully declared itself as the named item.

8.5.9 Viewer List

Opcode: 0x09

ContextName: String
ItemName: String
Viewers: vector of String

The Viewer list is the list of explicitly named viewers for the object named. All other viewers receive the default properties for that object. Note that the server does not know or care whether there are any actual differences between the properties in the default view cell and the properties in other view cells. The default-flag byte must be cleared (value of 0x00).

8.5.10 Fetch

Opcode: 0x0A

ContextName: String
ViewerName: String
ItemNames: vector of String
AndEnable: vector of bytes

Requests the properties of the named items, on behalf of the named viewer. The default-flag byte must be cleared (value of 0x00). Each name in ItemNames MUST be distinct.

A single fetch request takes place atomically. No client sees the result of a partially executed fetch.

If AndEnable is not a zero-length vector, then it MUST have the same number of elements as ItemNames. Each byte MUST be either the value 0x00 (ignore) or 0x01 (enable). For each byte that has the value "enable", the request enables change notifications immediately for the item named in the corresponding position of ItemNames (see "Enable", below).

8.5.11 Fetch Response

Opcode: 0x0B

ContextName: String
ViewerName: String
ObjectInfo: vector of ItemState

where ItemState is

ItemName: String
ItemProperties: vector of name-value pairs

Provides the properties of the named items, in response to a Fetch request. The default-flag byte must be cleared (value of 0x00).

8.5.12 Enable

Opcode: 0x0C

ContextName: String
ViewerName: String
ItemName: vector of String

Enables the delivery of change notifications. After this command is received, the server will deliver notifications to the named viewer whenever any of the named items in the named context change in a way that is visible to the viewer. The default-flag byte must be cleared (value of 0x00).

Depending on special server behavior (see Section 7) the server may deliver these notifications by a different channel rather than an existing SGAP transport connection. For example, the server may use a connectionless protocol such as UDP, or may open a server-to-client TCP connection to deliver the notification.

8.5.13 Disable

Opcode: 0x0D

ContextName: String
ViewerName: String
ItemNames: vector of String

Disables the delivery of notifications of changes. After this command is received, the server will not deliver any notifications to the named viewer from changes to the named items. The default-flag byte must be cleared (value of 0x00).

After issuing a disable request, a client must be prepared to receive some number of spurious (late) notifications and silently discard them.

8.5.14 **Creation**

Opcode: 0x0E

ContextName: String
ViewerNames: vector of String
ItemName: String
ItemProperties: vector of name-value pairs

Provides a notification to the named viewers that the named properties have been created as part of the named item (using a Create command).

The default-flag byte must be cleared (value of 0x00).

The ViewerNames are the viewers declared by this client that are receiving this notification, not all viewers receiving the notification. In particular, note that ViewerNames in this message is not identical to ViewerNames in the Create request that gives rise to this message.

Note also that not all of a client's viewers with access to the item will be named in ViewerNames: some of them may have not enabled notifications.

8.5.15 **Modification**

Opcode: 0x0F

ContextName: String
ViewerNames: vector of String
ItemName: String
ItemProperties: vector of name-value pairs

Provides a notification to the named viewers that the named properties of the named item have been modified (using a Modify command).

The default-flag byte must be cleared (value of 0x00).

The ViewerNames are the viewers declared by this client that are receiving this notification, not all viewers receiving the notification. In particular, note that ViewerNames in this message is not identical to ViewerNames in the Modify request that gives rise to this message.

Note also that not all of a client's viewers with access to the item will be named in ViewerNames: some of them may have not enabled notifications.

8.5.16 **Deletion**

Opcode: 0x10

ContextName: String
ViewerNames: vector of String
ItemName: String
ItemProperties: vector of String

Provides a notification to the named viewers that certain properties of the named item have been deleted (using a Delete command).

The default-flag byte must be cleared (value of 0x00).

The ViewerNames are the viewers declared by this client that are receiving this notification, not all viewers receiving the notification. In particular, note that ViewerNames in this message is not identical to ViewerNames in the Modify request that gives rise to this message.

Not all of a client's viewers with access to the item will be named in ViewerNames: some of them may have not enabled notifications.

8.5.17 OK

Opcode: 0x11

This is sent as an acknowledgement that a previous request was successfully completed, without errors. All requests except List Viewers and Fetch cause the server to reply OK if the request completed successfully.

8.5.18 Error

Opcode: 0xFF

ContextName: String
Code: Cardinal
StringData: Vector of String
Explanation: String

Provides a machine-processable error code, some data providing further context for the error, and a simple explanation that can be used for debugging. The explanations provided here are typical examples, rather than mandatory. Any such explanation provided by the server MUST NOT be presented to a user except as debugging information. Instead, the error code should be used to map to an appropriately localized error message.

Error code	StringData interpretation	Explanation

1	**Opcode value received,** as UTF-8 decimal representation	"Unrecognized Opcode"
2	**None**	"Not Authenticated"
3	**Depends on authentication** protocol	"Authentication Failed"
5	**Item name**	"Not Authenticated to Affect Item"
6	**Viewer name**	"Not Authenticated to Act As Viewer"
7	**Viewer name**	"No Such Viewer"
8	**Maximum bytes allowed in** value, as UTF8 representation	"Value Exceeded Server's Maximum Length"
9	**Context name**	"No Such Context"
10	**Allowed context names**	"Only These Contexts Allowed"
11	**Allowed item names**	"Only These Items Allowed"
12	**Allowed viewer names**	"Only These Viewers Allowed"

9. Acknowledgements

SGAP is a descendant of NSTP [NSTP, DPM] and has benefited from John Patterson's perspective in the design of that protocol. In addition, comments and criticism by Rob Ullmann, Dave Mitchell, Sandeep Singhal, Steve Foley, and Kamal Ayad have contributed greatly to improving the design. Naturally, any remaining errors are the responsibility of the author.

10. References

[DPM] Mark Day, John F. Patterson, and David Mitchell. "The Notification Service Transfer Protocol (NSTP): Infrastructure for Synchronous Groupware." Computer Networks and ISDN Systems, 1997.

[NSTP] Mark Day, John Patterson, Jakov Kucan, David Mitchell, Wei-Meng Chee. The Notification Service Transfer Protocol. Lotus Workgroup Technologies Technical Report 96-08, 1996.

[RFC-1034] Paul Mockapetris. "Domain Names -- Concepts and Facilities." RFC-1034, November 1987.

11. Changes Made in Revision 1

a. Changed version number changed from 0x05 to 0x85. The top bit will be set
for all future version numbers. This change enables SGAP and NSTP

packets to flow to a single server over a single connection.

b. Moved old Section 5 into Section 6. Old Section 6 rewritten, broken into smaller subsections.

c. Contexts explained.

d. Added SGAP:MIME type.

e. Added Init message; renumbered opcodes.

f. Modified Declare message to allow the default behavior even when using the long form of Declare.

g. Clarified that notifications are disabled and must be explicitly enabled.

h. Added new error codes.

12. Author's Address

Mark Day
Lotus Development Corporation
55 Cambridge Parkway
Cambridge, MA 02142
USA

Mark_Day@lotus.com