Abstract

The OOKIE project is a tool for allowing teams to virtually collaborate in real time. In particular, it allows groups of people to work together over a network by editing a shared document. Documents can include chats, text documents, and simple graphics. This document specifies the network protocol to be used by OOKIE-compliant collaboration tools.

1 Use Cases

There are several classes of users who would benefit from the use of a shared whiteboard. Some examples include:

Manny is a college student working on a project with a team of other students. They all have time to meet to discuss the project, but they would prefer to devote time to actual coordination rather than trying to find a meeting place and dealing with crowded libraries, etc. An on-line collaboration tool would allow Manny and his team members to focus on their project rather than on managing their team.

Paula is the manager and design lead for a software development team spanning the globe. While existing collaboration software such as wikis and web-based discussion fora allow Paula’s team to share information, there are times when interactive communication would greatly expedite their design process. However, traditional instant messaging systems are insufficient for their needs because they often need to use diagrams to convey system structure. An on-line collaboration tool would allow Paula’s team to discuss design decisions graphically without waiting for all team members to check a non-interactive shared resource.

Sally has friends around the country. She regularly uses the Internet to keep in touch with them, and occasionally she would like to communicate in more complex ways than are possible with chat-only systems. An on-line collaboration tool would allow Sally’s circle of friends to have richer on-line communications.

2 OOKIE Model

The OOKIE system deals with collaboration in a model similar to that used by most Internet chat systems:

- Users each assume an on-line identity (hereafter referred to as a login name). Login names may be associated with profiles containing information provided by the user.

- Users communicate with each other in separate concurrent sessions (hereafter referred to as rooms). Each shared document (for example, a shared text editing session or a collaborative whiteboard) has its own room. Each room also has an associated text chat, where users may post comments in real time, but the comments may not be modified after being sent to the room.
• Certain users (hereafter referred to as *admins*) have special privileges to restrict access to particular rooms. Admin status is particular to a specific room; that is, a user’s being admin in a particular room does not imply that that user is an admin in any other room. A user may become admin in one of two ways; the creator of a room is automatically made an admin, and an admin of a room may promote other users to be admins as well.

## 3 Network Protocol

The OOKIE protocol is simple and text-based. An OOKIE server listens on TCP port 3853 for OOKIE clients to connect, and the TCP connection is maintained until the user logs out. Messages (both to and from the server) all follow the general format of *command parameters* where the fields are delimited by whitespace. In order to accommodate the case in which a field may contain whitespace, a message may enclose a field in quotation marks. Further discussion of quoted fields is provided in Section 3.1. Each message’s end is indicated by a single newline character (in C++, the ‘\n’ character).

### 3.1 Quoted Fields

For fields which may contain whitespace, a message may enclose a field in quote marks (in C++, the ‘”’ character). A field quoted in this manner contains the literal value of each character until the closing quote (the quotes are not included in the field). This includes newlines and other whitespace. There are two (and only two) exceptions: in order to include a backslash (in C++, the ‘\’ character) the stream should include two backslashes in sequence. A quote mark as used to enclose the field can be included in the quoted field by preceding it with a backslash.

*Note:* The quote mark is only considered to be a delimiter if the field begins with the quote mark.

### 3.2 Messages

This section of the document is a listing of all the message types in the OOKIE protocol, with their purpose and required arguments. In the interest of brevity, the syntax for each is specified as follows:

- Literal text (messages are case-sensitive) is written with no special markup. For example, all message type strings are literal text.
- Free text is written with a dollar sign ($) prepended.
- A field that must contain the text form of a numeric value is written with a pound sign (#) prepended. All such fields are encoded in decimal.
- An ellipsis (…) at the end of a syntax specification indicates that the message may have different numbers of parameters in different situations; refer to the documentation for the specific message for more details.
- Square brackets ([ ]) indicate that a field is optional. This may be because a field has a specified default value or it may provide different functionality. Refer to the documentation for the specific message for more details.

#### 3.2.1 Client Messages

**SIGNON $LOGIN $PASSWORD**

This message identifies a user as signing on with the login name $LOGIN and the password $PASSWORD for authentication. If the user is already logged in when this message is sent, and the password matches the password stored by the server for that user, the server should close the older session. If the user does not
exist, then the server should create a new record for that user with the password provided, then respond as if the account had existed prior to the request. The server should reply with either a **DENIED** message to indicate that the login failed, or an **APPROVED** message to indicate that the login was successful. After a successful login, the server considers all messages from that client to be associated with the user whose login was provided until a **SIGNOFF** message is received. The user should also be considered to have signed off if the connection is dropped unexpectedly or if a successful login attempt for the same $LOGIN is received from another client.

*Note:* All other messages must be associated with a particular originating login. Therefore, the **SIGNON** message must be the first message sent by the client, and the server must ignore any other messages sent until a successful login.

**SIGNOFF**

This message alerts the server that a user is disconnecting. The server should reply with an **APPROVED** message.

**LISTUSERS [$ROOM]**

This message requests a list of currently logged-in users from the server. The server should respond with a **USERLIST** message. If the $ROOM field is included, only users currently in the specified room should be included in the response.

**LISTROOMS [$ROOM]**

This message requests a list of currently populated chat rooms. If no $ROOM is specified, the server should respond with a **ROOMLIST** message. If the $ROOM field is present, the server should respond with a **ROOMTYPE** message.

**LISTTYPES**

This message requests a list of room types supported by the server. The server should respond with a **TYPELIST** message.

**NEWROOM $ROOM $TYPE [$DOCUMENT]**

This message requests that the server create a new room with the name $ROOM and type $TYPE. If the $DOCUMENT argument is provided, then the server should interpret it as a sequence of commands according to the document type. These commands are not added to the document’s undo/redo stack, and should define the base document to be edited. The type should indicate what sort of document should be shared, and is specified by an OOKIE document specification distributed separately from this protocol specification. If the room is created, the current user will be the sole admin of the new room. The server should reply with either a **DENIED** message to indicate that the room creation failed, or an **APPROVED** message to indicate that the room creation was successful.

**JOINROOM $ROOM**

This message requests that the currently logged-in user be admitted into the room with the name $ROOM. The server should reply with either a **DENIED** message to indicate that the join failed, or an **APPROVED** message to indicate that the join was successful. The join may fail if the server does not support adding more users to a room, or if the admin of the room has denied it. The server should also send a message to all other users in the room to alert them of the new member.
**LEAVEROOM** $ROOM

This message informs the server that the current user is leaving a room. The server should reply with an **APPROVED** message to acknowledge the request, and should also notify the members of the room that the user has left.

**REMOVEROOM** $ROOM

This message requests that the server remove the room $ROOM. If the requester is an admin in the room, then the server should remove the room and respond with an **APPROVED** message; otherwise, the server should respond with a **DENIED** message. If a room is removed, the server should forward the **REMOVEROOM** message to all members of the room. Other than being removed by this command, the server should keep rooms indefinitely, even if they temporarily have no users.

**INVITE** $LOGIN $ROOM

This message requests that the user with login $LOGIN be invited to join the room $ROOM. The server should respond by replying with a **DENIED** message if the request is not allowed, or it should send an **INVITE** message to the specified user. Users may not invite other users to a room that they themselves are not in. (That is, if user A is in not in room X, he may not invite user B to join room X.) However, the protocol does not prevent users from cruelly inviting others to rooms they may not join.

**SETINFO** $INFO

This message informs the server of the information the user would like to use for his or her profile. It is up to the server to impose limits on the length of this information. The server should reply with an **APPROVED** message to acknowledge this request.

**GETINFO** $LOGIN

This message requests the user profile for the login $LOGIN. The server should reply with an **INFO** message. Clients should be prepared to handle HTML formatting in the profile.

**MESSAGE** $ROOM $TEXT

This message tells the chat server that the user would like to post the message $TEXT to the text chat associated with the room $ROOM. The server should reply with a **DENIED** message to indicate that the user does not have permission to send messages to the room, or it should add the sender’s $LOGIN to the message and forward the message to all members of the room. The client should not display the message until the server has replied (this ensures that all clients will have the messages in the same order, and also keeps the client from having to remove denied messages after having already displayed them).

**APPEND** $ROOM #REV ...

This message adds a new item to the shared document for the room $ROOM. The #REV field indicates the latest revision of the document that the client knows about when the message is sent. The parameters for the item to be created are specified differently based on the type of the document. The server should reply with a **DENIED** message to indicate that the user does not have permission to send messages to the room, or it should correct the message for any intervening edits, assign the appropriate version number, and forward it to all members of the room. Document type-specific modifications to the message may also be made; these
are specified in the protocol extensions for each document type. The client should not display the change until the server has replied (this ensures that all clients will have the messages in the same order, and also keeps the client from having to remove denied messages after having already displayed them).

REMOVE $ROOM #REV ...
This message removes an already existing item from the shared document for the room $ROOM. The #REV field indicates the latest revision of the document that the client knows about when the message is sent. The parameters used to specify the item to be removed are specified differently based on the type of the document. The server should reply with a DENIED message to indicate that the user does not have permission to remove the item, or it should modify the message according to the protocol extension for the current document type, set the revision number appropriately, and forward the message to all members of the room. The client should not display the change until the server has replied.

MODIFY $ROOM #REV ...
This message modifies an already existing item in the shared document for the room $ROOM. The #REV field indicates the latest revision of the document that the client knows about when the message is sent. The parameters for the modification are specified differently based on the type of the document. The server should reply with a DENIED message to indicate that the user does not have permission to modify the item, or it should modify the message according to the protocol extension for the current document type, set the revision number appropriately, and forward the message to all members of the room. The client should not display the change until the server has replied.

MODIFY $ROOM UNDO $LOGIN
This message requests that the server undo the last operation performed by the user with login $LOGIN in the room $ROOM. This modification increments the revision number for the document. The server should reply with a DENIED message if the undo stack is empty or the requesting user does not have permission to undo the operation; otherwise the server should forward the message to all members of the room (including the requester).

MODIFY $ROOM REDO $LOGIN
This message requests that the server redo the last operation undone by the user with login $LOGIN in the room $ROOM. This modification increments the revision number for the document. The server should reply with a DENIED message if the redo stack is empty or the requesting user does not have permission to undo the operation; otherwise the server should forward the message to all members of the room (including the requester).

Note: The terms undo stack and redo stack refer only to a common model for undo/redo operations. The implementation of the undo/redo history is up to individual server designers.

PERMS $ROOM $LOGIN [$ACL]
If the $ACL field is omitted, this message requests the permission information for the user with login $LOGIN in the room $ROOM. The server should reply with a PERMS message to only the requesting user.

If the $ACL field is included, this message sets the permissions for the user with login $LOGIN in the room $ROOM. The server should reply with a PERMS message indicating the user's new permissions to both the
requesting and the affected user if the requesting user is an admin for the room, or it should send a DENIED message to only the requester if the requesting user is not an admin. The server should also send a DENIED message if an admin attempts to remove his or her own administrative status. The $ACL$ field should consist of four characters in order, each either a specific letter or a dash (−) to indicate the privilege is denied. In order, the characters are:

A admin status
W write permission; that is, the ability to add/modify/remove items in the document
R read permission; that is, the ability to join the room at all.
P protection. Content added to a document by a protected user can only be modified by the creator or an admin.

For example, to request that the user Bob have full permissions in the room OOKIE, the message would be:

PERMS OOKIE Bob AWRP

### 3.2.2 Server Messages

**DENIED $REQUEST [$NOTE]**
This message is the response to a request that is denied due to security restrictions (or implementation-imposed limits such as insufficient memory to spawn a new room). $REQUEST should be the message type that was denied. $NOTE is an optional message that should be informative to normal users (that is, people familiar with the features of the client but not with the details of the protocol). The server should guarantee that DENIED messages will be sent in the same order as the corresponding requests for a particular user.

**APPROVED $REQUEST [$NOTE]**
This message is the response to a request that is approved, but needs no further information (such as an account creation request). $REQUEST should be the message type that was approved. $NOTE is an optional message that should be informative to normal users (that is, people familiar with the features of the client but not with the details of the protocol). The server should guarantee that APPROVED messages will be sent in the same order as the corresponding requests for a particular user.

**ROOMLIST ...**
This message is the response to a LISTROOMS request with no room specified by the client. Aside from the message type, each field is the name of a room on the server. The server may exclude rooms from the list if the requester does not have permission to join them, but all rooms that the client may join must be listed.

**ROOMTYPE $ROOM $TYPE**
This message is the response to a LISTROOMS request with a room specified by the client. The $ROOM field should indicate the name of the room whose type was requested, and the $TYPE field should indicate that room’s type.

**REMOVEROOM $ROOM**
This message is used to notify members of a room that the room is being removed from the server.

**INVITE $LOGIN $ROOM**
This message is used to alert a user to the fact that the user with login $LOGIN has invited him or her to
the room $ROOM.

TYPELIST ...
This message is the response to a LISTTYPES request from a client. Aside from the message type, each field is a possible type that the client may specify when creating a room.

INFO $LOGIN $PROFILE
This message is the response to a GETINFO request from a client. The $LOGIN field should be the login name of the user whose profile info is retrieved, and the $PROFILE field should contain the profile data provided by the user.

PERMS $ROOM $LOGIN $ACL
This message is the response to a request for, or modification of, permissions information. The format is the same as the client message format for setting permissions.

MESSAGE $ROOM $LOGIN $TEXT
This message notifies members of the room $ROOM that a message has been posted to the chat session associated with the room. $LOGIN is the login name of the user that sent the message.

APPEND $ROOM $LOGIN #REV ...
This message notifies members of the room $ROOM of a new item added to the shared document for the room. The #REV field indicates the revision number of the document after the addition. The parameters for the item to be created are modified based on rules defined separately for each document type. $LOGIN is the login of the user that requested the change, and #REF is a numeric identifier for the item being added. Identifiers should be added sequentially and not reused in a single chat (even if the previous item with a particular number has been removed from the document).

REMOVE $ROOM $LOGIN #REV ...
This message notifies members of a room that an already existing item has been removed from the shared document for the room $ROOM. The parameters used to specify which item has been used are specified separately for each document type. The #REV field indicates the revision number of the document after the removal.

MODIFY $ROOM $LOGIN #REV ...
This message notifies members of a room that an already existing item with reference number #REV has been modified. The parameters for the modification are specified differently based on the type of the document, and may be modified by the server to accommodate for changes made between the time the request leaves the client and the time that it is processed by the server. The server should reply with a DENIED message to indicate that the user does not have permission to modify the item, or it should forward the message to all members of the room.
3.2.3 Unrecognized Messages

OOKIE servers and clients should respond to unrecognized messages by replying with a message of the format: UNKNOWN $MESSAGE

where $MESSAGE is the first field (ie, the message type) for the unrecognized message. This applies to malformed messages as well.

4 OOKIE Extension 1: Text Document

This section specifies an OOKIE protocol extension for sharing a plaintext document. Because of the lack of structure in such a document, this is a very simple extension; possibly the simplest possible useful extension to OOKIE. The extension type string for text documents is TEXT.

4.1 APPEND Commands

There are no APPEND commands for text documents.

4.2 MODIFY Commands

MODIFY $ROOM #REV REPLACE #LINESTART #CHARSTART #LINEEND #CHAREND $TEXT

This command replaces the range of text that begins at the line and character specified by #LINESTART and #CHARSTART and ends just before the line and character specified by #LINEEND and #CHAREND with the text in the field $TEXT. Line and character numbers are zero-based indices. If the #LINESTART and #CHARSTART are the same as #LINEEND and #CHAREND then this command inserts $TEXT before that position. If the current revision of the document is greater than the revision listed by the client, then an appropriate transformation of the region should be made (for example, if a line is inserted before #LINESTART then both #LINESTART and #LINEEND should be incremented. This command is to be used for insertion, replacement, and deletion.

Note: An empty field can be sent by using two adjacent quotation marks.

MODIFY $ROOM #REV HIGHLIGHT #LINESTART #CHARSTART #LINEEND #CHAREND

This command specifies that the current user is attempting to draw focus to a range of text specified in the same way as discussed in the description of the REPLACE command. Since the text document format does not support division of a document into elements, highlighted sections are used instead. That is, if a user with the protect attribute has a section of text highlighted, the server should refuse requests to modify that text. Users should only be allowed to select one region of text at a time, however.

4.3 REMOVE Commands

There are no REMOVE commands for text documents.

5 OOKIE Extension 2: Drawing Document

This section specifies an OOKIE protocol extension for sharing a drawing. The document type string for drawings is DRAWING. Drawings are divided up into individual elements, each consisting of some shape (such as a rectangle or a freehand drawing). For the purposes of drawings, an internal canvas should be used with dimensions of 10000x10000 and coordinates should be specified as integers in the range [0..10000]. For the command specifications in this section, the prefix % is used to indicate a coordinate in the format of two decimal values separated by a comma (in C++, the ‘,’ character). Each item has four attributes: outline
color, fill color, size, and position. The outline color for newly added elements should be black, the fill color should be transparency, and the size and position are specified by the APPEND command.

5.1 APPEND Commands

APPEND $ROOM #REV OVAL %BOTTOMLEFT %TOPRIGHT
This command creates an oval that fills the bounding box whose bottom left corner and top right corner are %BOTTOMLEFT and TOPRIGHT, respectively. The server should respond by prepending the keyword OVAL with a numeric identifier that is unique within the session. The format for this reply message, therefore, is

APPEND $ROOM $LOGIN #REV #ID OVAL %BOTTOMLEFT %TOPRIGHT

APPEND $ROOM #REV RECT %BOTTOMLEFT %TOPRIGHT
This command creates a rectangle whose bottom left corner and top right corner are %BOTTOMLEFT and TOPRIGHT, respectively. The server should respond by prepending the keyword RECT with a numeric identifier that is unique within the session. The format for this reply message, therefore, is

APPEND $ROOM #REV #ID RECT %BOTTOMLEFT %TOPRIGHT

APPEND $ROOM #REV LABEL %BOTTOMLEFT $TEXT
This command creates a text label whose bottom left corner is specified by the %BOTTOMLEFT field. The server should respond by prepending the keyword LABEL with a numeric identifier that is unique within the session. The format for this reply message, therefore, is

APPEND $ROOM #REV #ID LABEL %BOTTOMLEFT %TEXT

APPEND $ROOM #REV SQUIGGLE %POINT ...
The ellipsis for this command should be replaced by a sequence of coordinate sets of arbitrary length. The command creates a set of line segments (all considered part of the same item) connecting the points. The first and last points are not connected. The server should respond by prepending the keyword SQUIGGLE with a numeric identifier that is unique within the session. The format for this reply message, therefore, is

APPEND $ROOM #REV #ID SQUIGGLE %POINT ...

5.2 MODIFY Commands

MODIFY $ROOM #REV #ID OUTLINECOLOR #R #G #B
This command sets the outline color of the element whose numeric id is #ID to the RGB color specified by #R, #G, and #B. The colors are specified in the range [0..255] where 0 is the darkest and 255 is the brightest possible value for each color.

MODIFY $ROOM #REV #ID FILLCOLOR #R #G #B
This command sets the fill color of the element whose numeric id is #ID to the RGB color specified by #R, #G, and #B. The colors are specified as in the OUTLINECOLOR command.

MODIFY $ROOM #REV #ID RESIZE %BOTTOMLEFT %TOPRIGHT
This command resizes and moves the element to fill the specified bounding box. This command should be used for both resizing and moving elements.
MODIFY $ROOM #REV #ID SETTEXT $TEXT
This command replaces the text for the label with the specified #ID. If the item is not a text label, the server can either ignore the request or send a DENIED message in response.

5.3 REMOVE Commands
REMOVE $ROOM #REV #ID
This command removes the item specified by the numeric id #ID.

6 OOKIE Extension 3: Weekly Calendar
This section specifies an OOKIE protocol extension for sharing a weekly calendar. The document type string for a shared calendar is CALENDAR. For the command specifications in this section, the prefix @ is used to indicate a date and time in the format of YYYY-MM-DDThh:mmTZD as specified in ISO Standard 8601. (More details can be found at http://www.iso.org/iso/en/prods-services/popstds/datesandtime.html or http://www.w3.org/TR/NOTE-datetime.)

6.1 APPEND Commands
APPEND $ROOM #REV $DESCRIPTION @START @END
This requests that a new event be added to the calendar. $DESCRIPTION is a text description of the event. @START and @END are the start and end times of the event, respectively. If the request is approved, the calendar should add the sender’s login name and a numeric identifier to the request and forward it to all members of the chat. The format of the message from the server will be
APPEND $ROOM #REV #ID $LOGIN $EVENT @START @END

6.2 MODIFY Commands
MODIFY $ROOM #REV #ID START @START
This message requests that the start time of the event with numeric id #ID be replaced with the specified value.

MODIFY $ROOM #REV #ID END @END
This message requests that the end time of the event with numeric id #ID be replaced with the specified value.

MODIFY $ROOM #REV #ID DESC
This message requests that the description for the event with numeric id #ID be replaced with the specified value.

6.3 REMOVE Commands
REMOVE $ROOM #REV #ID
This message requests that the event with numeric id #ID be removed from the calendar.