Project	**IEEE 802.16 Broadband Wireless Access Working Group <http://ieee802.org/16>**
Title	**Documentation of a Comment Resolution Process**
Date Submitted	**2000-08-02**
Source(s)	Brian Petry 3Com 12230 World Trade Dr. San Diego, CA 92128 Voice: 858-674-8533 mailto:brian_petry@3com.com
Re:	Procedural documentation
Abstract	This document describes a process used to accept and resolve comments on 802.16 documents. The process is designed to be simple, flexible and fairly automated. This is version 2.0 (802.16-00/16r1), with these changes over version 1.0 (802.16-00/16): • A description for managing revisions using FrameMaker. • Updated MS-Access database with more compact reports and a few minor enhancements.
Purpose	This contribution could be used by 802.16 task groups, or other 802 groups, to help administer a formal document revision, to study the process and *make improvements*.
Notice	This document has been prepared to assist IEEE 802.16. It is offered as a basis for discussion and is not binding on the contributing individual(s) or organization(s). The material in this document is subject to change in form and content after further study. The contributor(s) reserve(s) the right to add, amend or withdraw material contained herein.
Release	The contributor grants a free, irrevocable license to the IEEE to incorporate text contained in this contribution, and any modifications thereof, in the creation of an IEEE Standards publication; to copyright in the IEEE's name any IEEE Standards publication even though it may include portions of this contribution; and at the IEEE's sole discretion to permit others to reproduce in whole or in part the resulting IEEE Standards publication. The contributor also acknowledges and accepts that this contribution may be made public by IEEE 802.16.

Patent Policy and Procedures	The contributor is familiar with the IEEE 802.16 Patent Policy and Procedures (Version 1.0) <http://ieee802.org/16/ipr/patents/policy.html>, including the statement "IEEE standards may include the known use of patent(s), including patent applications, if there is technical justification in the opinion of the standards-developing committee and provided the IEEE receives assurance from the patent holder that it will license applicants under reasonable terms and conditions for the purpose of implementing the standard." Early disclosure to the Working Group of patent information that might be relevant to the standard is essential to reduce the possibility for delays in the development process and increase the likelihood that the draft publication will be approved for publication. Please notify the Chair <mailto:r.b.marks@ieee.org> as early as possible, in written or electronic form, of any patents (granted or under application) that may cover technology that is under consideration by or has been approved by IEEE 802.16. The Chair will disclose this notification via the IEEE 802.16 web site <http://ieee802.org/16/ipr/patents/letters>.

Table of Contents

1 Documentation of a Comment Resolution Process

Brian Petry

3Com

Version 2.0

2 Introduction

This document describes a process for accepting and resolving comments on documents. It is intended for people who undertake the administration of a comment/resolution process and/or seek to improve on the process described herein. 802.16 has used this process successfully for several rounds of comments on functional requirements documents. But to date, it has not been used for working-group ballots.

Section 5 describes the software tools required to enable this process. Appendices provide examples, scripts, etc.

Accompanying this document, the reader should find electronic files that include:

- A README.txt file (look at this one first)
- An example call-for-comments
- The comment submittal form
- An empty MS-Access Database
- An example populated MS-Access Database
- An example of on-line comment resolution instructions
- An example on-line ballot form
- Perl scripts for manipulating submitted comment forms and ballot forms

The reader should check the 802.16 web page (http://ieee802.org/16) for potential updates to this document and the associated files.

2.1 Submittal Approach

The Comment submittal process uses text-based email. A text format yields a high degree of flexibility, and is a format easily readable, e-mail-able, and exportable, to all computing platforms, databases, etc.

An attractive alternative to text-based email is on-line comment submittal through a web page. Web-based submittal seems attractive from the viewpoint of the commentor: submittal is structured and less error-prone. Also, a database "back-end" to the web page may be employed that offers a high degree of automation. However, to surpass the level of success as text-based email submittal, web-based submittal has some administrative costs and complexities that must be addressed:

- The expense of a database back-end system. To achieve full automation, a database can be linked to the web page so that comment forms can be submitted directly into a database. Such automation requires an investment of time and money to set up, test, administer, etc. The database back-end could be used for commentors to review all of their comments for consistency prior to issuing a final “commit.” It’s easy for a commentor to do this with an email based submittal system, because all of the comments can be kept in one file that the commentor edits. And when the commentor is satisfied, can email all the comments at once.
- Authentication of commentors. Commentors must be somehow be authenticated (e.g., with secure password/etc.) before comments can be accepted. This adds complexity and expense (administration time) to the process.
- Some commentors may find using their favorite editor or word-processor may be more effective to gather their comments, rather than submitting them one-by-one on a web page using a text-based form. The point-and-click, cut-and-paste user interface of a web form can be more burdensome than the email-based submittal process.

So, the process described in this document strives to achieve simplicity, flexibility and automation offered by text-based email comment submission. Further, it uses tools that are free or commonly available to administer the process. But if someone can afford to develop a web-based system that supports “features” mentioned above, so as to surpass the effectiveness of the text-based email approach, that would be welcome.

3 Process Outline

An outline of the process follows.

Prior to the next meeting

- Issue a call-for-comments on a specific document
- Commentors email their comments (in a strict format) to the group's reflector.
- An editor saves all comments in text file(s)
- An editor reviews all comments and edits them into the document, with change bars and red-lines
- An editor imports all comments into an MS-Access database
- An editor publishes a report of all comments received (e.g., a database report)

Optionally, comment resolution may proceed via email

- Issue an announcement and instructions
- Commentors fill-out text ballots (in a strict format) and email them to an editor or ballot collector
- Verify that submitters meet the membership requirements for balloting

- Compile results (using an automated tally program)
- Update the document with appropriate resolutions
- Publish a new document and voting results

At the comment resolution meeting(s)

- Comments are resolved, one by one, with editor(s) projecting on the overhead screen: A view of the comment database, the original document, and the document version being edited.
- For each comment, an editor, using the database, tracks how each comment is resolved by the group
- For each comment, an editor makes effects changes to the document on-the-fly
- For each comment, a secretary records motions, votes, etc. for the minutes

After the comment resolution meeting(s)

- An editor publishes a comment resolution report (from the database). This can be included in the minutes
- Ideally, after the session, the editor(s) have:
    - A new version of the document that is very close to being publishable
    - A report of how all comments were resolved

4 Details

This section describes some detail behind each of the steps in the process.

4.1 Prior to Meeting

4.1.1 Issue a call-for-comments

This is the critical step that kicks-off the comment/resolution process. The call-for-comments must be clear and precise. It contains:

- An overview of the process: what commentors need to know
- Links to appropriate documents, including the document under question and the text-based comment submittal form
- Instructions for commentors on how to submit comments
- A deadline by which comments must be received

An example of a call-for-comments is given in appendix A.

4.1.2 Commentors email their comments

A strict format for comments provides an interface that is fair to all commentors and less error-prone for the editor(s). Furthermore, it provides a format that is convenient and suitable for import into a database (MS-Access). The comment format is text-based with clear delimiters that separate each field of a comment:

First and last name of commentor

Starting page number: The document page number to which the comment applies

Starting line number: The document line number to which the comment applies

Type of comment: "Technical" for comments that cover technical issues or probably require debate, "Editorial" for comments that will not require debate.

Detailed description of change, add, or delete. The commentor must specify a specific and detailed change to the document.

Reason for change. The commentor must provide a descriptive reason for the change. This is especially important for technical or controversial changes.

Commentors fill-out one form for each comment, concatenating the forms together in an email message and sends the comments to the group's reflector.

4.1.3 An editor saves all comments

An editor saves all comments in text file(s), verifying that they are properly formatted. If comments are not properly formatted, the editor should reject the comments, requesting the commentor to reformat the comments in the specified form..

Even though the email reflector may automatically archive the comment messages, the editor should be careful to back-up received comments to prevent loss.

Typically, a commentor will submit multiple (many) comments in one email message. The editor should save a commentor's message, containing multiple comments, in a separate text file. Furthermore, email headers and footers (e.g., email signatures, etc.) must be removed. Later, all comment files can be concatenated together into one file.

4.1.4 Edit-in the comments

An editor reviews all comments and edits them into the document, with change bars and red-lines. Some comments will be duplicates, and some comments will overlap and conflict with other comments. In the document, the editor should note clearly when such overlaps and conflicts occur. Also, it is convenient for the commentor's name and the type of comment (T for technical, E for editorial) to appear in document. These notations will simplify the resolution process at the next meeting. The editor should take care to use the change bars wisely, with a small number of change "transactions." For instance, if the commentor wants to replace text, (i.e., using MS-Word) one change transaction that both deletes the text and adds the requested text will help simplify on-the-fly acceptance or rejection during the meeting.

The editor may publish the edited document prior to the next meeting so people can have it in front of them during face-to-face comment resolution. But sometimes, this can lead to confusion, because the commentors then have two documents to hassle with: oen they submitted their comments to, and another that has their change edited-in.

Note that MS-Word handles change bar mechanisms, and on-the-fly acceptance or rejection of changes fairly well. MS-Word allows the user to simply highlight, or click on a change and right-click to either accept or reject the change. So, during the meeting, an editor can make efficient use of meeting time and quickly dispose of the comment and go on to the next one. However, FrameMaker should be a much more effective tool overall (it's what the IEEE staff editors require). The next section describes instructions for an effective revision marking mechanism in Frame.

4.1.5 FrameMaker revision marking

If the Frame template you are using doesn't already have them, add two new character style tags, called "PendingInsert" and "PendingDelete." These will be used to conveniently mark text that represents changes requested by commentors.

4.1.5.1 Create revision tags

[Format->Characters->Designer]
Type "PendingInsert" into the Character Tag field.
Select styles as "Color: Blue," "Underline," and "Change Bar." Also set the value "As is" for all other font parameters "Family," "Size," "Angle," "Weight," and "Variation."

[Commands->New Format]
Select "Store in catalog" and not "apply to selection."

[Create]

As with the PendingInsert tag, PendingDelete is similar…

Type "PendingDelete" into the Character Tag field.
Select styles as "Color: Red," "Strikethrough" and "Change Bar." Also set the value "As is" for all other font parameters "Family," "Size," "Angle," "Weight," and "Variation."

[Commands->New Format]
Select "Store in catalog" and not "apply to selection"

[Create]

4.1.5.2 Editing

While editing, leave open a the character format catalog [Format->Characters->Catalog]. You’ll see two new entries there for “PendingInsert” and “PendingDelete.” As you edit the document with proposed changes, highlight text and tag it with the new tags. When you’re done, you’ll have a document with changebars, red overstrike deletions and blue underline insertions.

4.1.5.3 In front of the committee

While accepting and rejecting changes in front of the committee (using an lcd projector) the following steps should help quickly identify changes and accept/reject them.

Again, leave open the character format catalog [Format->Characters->Catalog].

Use the Find/Change feature to quickly locate a PendingInsert or PendingDelete: [Edit->Find/Change]. Select “Character Tag:” in the “Find” field, and type “Pending” in the next field to the right. This will cause the find operation to match either the PendingInsert or PendingDelete tag.

Highlight the change so it stands out for the committee. If they decide to keep the pending text, just select “Default Font” from the character format catalog. If the committe decides not to keep the text, delete it.

The Find Next feature [Edit->Find Next] (use the keyboard shortcut: “Alt-e n”) to locate the next pending change.

4.1.6 Convert comments into an importable format

Importing comments into an MS-Access database takes a few steps, and is slightly problematic, so also expect some iterations of import attempts. Despite the strict comment submittal form, inevitably some commentors will mess-up by accidentally changing delimiters, etc.

First, some common syntax errors in the comments may need to be fixed. A Perl script, “fix.pl” (see appendix) puts blank lines back into comment files that a commentor may have deleted. Other syntax errors may need to be fixed manually.

Second, the comment files from different commentors should be concatenated together in one big text file. This simplifies the next step.

Third, the one big comment file must be run through another Perl script, “cimport.pl” which reformats the comments into a delimited format understood by MS-Access. The editor specifies an initial comment index and date as parameters to the Perl script. The comment index is a “key” to the database. Specifying a starting index other than 0 (or 1) allows a batch of comments to be imported into an existing database. For example, if 100 comment (0..99) are already in the database, the editor can use index 100 to import another batch. The date is added to each comment record at the import step to record the date the comment was added to the import. The date is useful if comments need to be identified by date, for instance, if later a new batch of comments is accepted from commentors. The date helps to separate multiple import batches later, if necessary. Also during this step, the script may detect syntax errors in the comment file. A common problem here is that the commentor did not use a decimal-only number in the page and/or line fields. These problems need to be fixed manually before continuing.

The import file generated from this step must be a DOS text-formatted file (with CR and LF on the end of each line), and have the typical Microsoft ".txt" file name extension.

Here are some more notes concerning the database and import functions:

- If the database structure is changed (fields are added or deleted), the cimport.pl script must be changed to accommodate the new database features. When this happens, some dry-runs are necessary, with sample comments and an empty database. Happy debugging.
- One field in the database is "hard-coded" in the comment import script: the document number in question. This field is present in the database in case one database should be used for different documents, but comments need to be managed together in one database. So far, 802.16 has not used this database feature.
- Here's one way this process could be improved. A slight problem is that MS-Access can't sort by multiple fields in data entry forms, such as the comment resolution screen which is displayed in front of the committee. So, you can sort the view of the database by page number, but not by page number and then line number within the page. This makes it a slight hassle to locate the "next comment to resolve." A new Perl script could pre-sort all comments by page # and line # so that when they are imported into the database, the natural order of the resulting database table is the order in which a comment is typically resolved. The comment numbers are thus sequential and this natural order will help from fumbling around searching for the "next comment to resolve" in front of the committee.

4.1.7 Import the comments into MS-Access

MS-Access must import the file generated from the previous step.

Comments are imported into an empty or non-empty database.

If this is the first comment import, an empty database must be used: find a file called "Empty Comment Database.mbd". The empty database has the database tables, data-entry screen forms, report formats, etc. already defined.

Whether starting from an empty, or non-empty database, make a copy of the database file (.mbd) before attempting an import because the import process is somewhat error prone, and is not "undoable."

Here's how to navigate the MS-Access import "wizard (jester):"

[File]->[Get External Data]->[Import]

A file selection dialog appears. Choose ".txt" from the [Files of Type] pull-down menu. Pick the import file and click:

[Import]

Now, the “Import Text Wizard” should appear. Select the [Delimited] option. Click [Next->].

Choose [Other] delimiter and enter the vertical bar (pipe) symbol: |.

Select [Text Qualifier] and enter the “at” symbol: @.

Select [First Row Contains Field Names].

At this point, a display in the dialog box should display the first 20 or so comments that will be imported. The comment entries should have a reasonable-looking format. If not, something is wrong, and it must be fixed before continuing.

Click [Next->].

Select import [In an Existing Table] and select “comment” (that should be the only option). Click [Next->].

Click [Finish] to import the file. If successful, a dialog box should appear that says something like: “Finished Importing File…” At this point, however, MS-Access may have found further errors in the import file, but will likely give the operator the option of continuing with the import anyway. To find out exactly what the errors are, continue with the import and MS-Access will generate a table, including record numbers, of comments that have problems. These problems need to be fixed before continuing.

A few iterations of the above steps are usually necessary before comments are successfully imported. If the import fails, delete the database file and revert to a new copy of the database (empty or non-empty). Then edit the big comment file to resolve any errors that were reported by MS-Access and go back to step 4.1.6.

4.1.8 Generate a comment report

An editor publishes a report of all comments received. This is a simple matter of using the MS-Access “Report” features to format a document, and a few report templates are already defined in the empty database to which comments were originally imported.

There are two purposes for the comment report:

To capture and archive all comments in a simple, non-database format.

To facilitate comment resolution at the next meeting.

To achieve both requirements, the report should group technical and editorial comment separately, and then sort comments by page # and line# in the document. This allows editorial comments to be conveniently reviewed and accepted into the document as a “batch,” and allows all comments to be reviewed in the order in which they apply to the document.

And here’s some more minutiae. The report MS-Access generates is a “printer” format. To turn this into an archivable document, it needs to be a “pdf” file. And once the MS-Access pdf-format report is generated, a cover page (e.g., an 802.16 contribution template) should be prepended to the report.

At this point in the process, the comment database report and the marked-up document can be published.

4.2 On-Line, Between Meetings (optional)

Optionally, especially if time is limited to publish the document in question, comments may be resolved between meetings. Also, if a large quantity of comments are expected on a document, many can be dealt with on-line thus saving meeting time. Further, if there is enough time left between the end of on-line comment resolution and the next meeting, another round of comments may be accepted on the document. Then, at the next meeting, the on-line comment “resolutions” can be finalized and another round of comments processed face-to-face.

One note of concern with on-line voting is that a person’s membership in the 802 working group could possibly be affected by not participating. It is important to review the working group rules and inform the group how on-line voting may affect membership.

Another concern is how to handle votes for comments marked “technical” versus those marked “editorial.” The task group (or working group) may require that technical comments need a 75% majority to accept into the document, and simple majority for editorial comments.

It is highly likely that not all comments can be adequately resolved by this on-line process because of majority concerns or “accept-but-only-if-modified” votes. The “accept-if-modified” vote allows for the case that the comment is almost acceptable, but would require a “friendly amendment” to be accepted. Thus, the accept-if-modified votes and the suggested friendly amendments can be used as input for the next face-to-face meeting.

802.16 has used on-line resolution only once, for the 802.16.1 Functional Requirements. As with comment submittal, on-line comment resolution utilized text base email for “vote” submittals. It worked reasonably well, but an automated web-based ballot may be more practical. Email submittal required less set-up time. Here are the steps 802.16 used:

4.2.1 Publish instructions

An editor publishes instructions for on-line comment resolution. It is a form of on-line voting. Example instructions appear in the appendix. The instructions specify who is invited to participate (e.g., members, observers, etc). The instructions point to a text-based form that voters must use to submit their opinions on comments. The form (an example is given in the appendix) lists all comments, one per line, with just the comment number (the database index), a vote field (possible votes are: accept, accept-if-modified, reject or abstain)and a reason field (for the reject and accept-if-modified cases). The form should be pre-filled-out with "accept" as the default vote. This default allows voters to quickly accept most comments, and only do additional work if they need to abstain or reject. It basically leaves the benefit of the doubt to the commentor.

The instructions must specify a strict deadline for submitting ballots. Also, the group must decide whether the votes must be submitted to the group's email reflector or only to an editor.

4.2.2 Voters submit filled-out ballot forms.

As in the comment submittal phase, voters submit their ballots via email according to the instructions. As ballots come in, the editor must reliably archive the electronic ballots.

4.2.3 Compile results and update database

The editor runs the text ballots through a Perl script (votecount.pl, see appendix) that tabulates results and generates a report of the "reason" fields. The editor also edits a new version of the document, incorporating all resolved comments.

The editor must also update the comment database to reflect any resolved comments. For comments that were not resolved (e.g., no clear majority or a majority of accept-if-modified votes), the comment database should retain these for resolution at the next face-to-face meeting. Accept-if-modified reasons can be incorporated into the database if necessary, and the next version of the document even red-lined with such suggestions.

4.2.4 Publish reports for next meeting

The report from the previous step, the resolution status of the online ballots, and a new version of the document, that reflects resolved comments, should be published. The group must decide how widely the individual votes should be published: to the public, to members only or to members and observers.

One issue to carefully manage is that comments the group might want to make a vote of confirmation at the next meeting whether the comments were resolved adequately. To help this eventuality, the editor can effect the resolved changes in the document with a different color, and perhaps use a special field in the comment database that indicates if a comment's resolution status is subject to such confirmation.

4.2.5 Publish another call for comments (optional)

Goto step 4.1.1. Note that when the editor imports a new batch of comments, the same database which holds the unresolved comments, can be used. The new comments are imported "on top of" the unresolved comments by specifying a starting comment index when the Perl script (cimport.pl, see appendix) is used.

4.3 At the meeting

During a comment resolution session, 802.16 has learned some useful conventions to run the process smoothly and efficiently---most of the time. Other groups, such as 802.3 and 802.11 have resolved very large numbers of comments at their meetings to resolve working group ballots (working group ballots are perhaps more formal than the comments described in this document), and may provide further advice for handling comment resolution at the meeting. For instance, they typically divide comments by relevant document sections and run parallel sessions to resolve them.

The meeting requires at least 3, and maybe 4 administrators:

1) A chair, who leads the session and administers the rules of order
2) An editor, who "drives" the overhead display(s) with a database view, a "new document" view and an old document view. If possible, multiple displays can be utilized to see the database and the document simultaneously.
3) A secretary, who keeps accurate minutes of the session and the resolution status of all comments.
4) Optionally, someone can serve as a "queue manager," who keeps track of who talks next, and any imposed discussion time limits.

4.3.1 Database View

The database view, that the editor(s) project, utilizes a form (it is already defined in the empty database along with this document) that projects one comment at a time on the screen. Shown are the submitter's name, page number, line number, resolution status, change description, reason, comment type (technical or editorial), item number, and notes. Some comments about "driving" this form follow.

The most useful features of the form involve sorting and filtering. These features are called-up by right-clicking in the field of concern (field to be sorted or filtered). Sorting is semi-useful. You can sort by one field only; you can't sort by one field, then another field within that sort, etc. The worst effect of this limitation is that the database view cannot sort comments by page # and then within the page, by line #. So, when the editor scrolls between comment records (using the page-up, page-down keys), it's likely that successive records won't be in line-order on the page. This results in slowing down the face-to-face meeting time, as the editor must scroll around through all comments on a page in search of a comment in question. Thus, the process could be significantly improved if somehow the multi-sort limitation could be fixed. Note that MS-Access Reports can easily perform such hierarchical sorts. The other useful feature, filtering, can be used to remove from the displayed comment list, certain comments subject to the filter. For instance, it is convenient to remove all but unresolved comments from display.

As the meeting proceeds, the editor works with the chair and secretary to locate the next comment for debate and discussion, display appropriate comments and record comment numbers in the minutes. Then, as comments are resolved, updates the comment database on-the-fly by updating the "Resolution Status" field and the "Notes" that captures the conclusions of the task group regarding the comment. The "Resolution Status" is one of:

- Unresolved: No resolution on this comment, yet (initial/default state).
- Accepted: The comment has been resolved without modification.
- Accepted-Modified: The comment has been resolved, but the group modified it in some way. The "Notes" field of the comment record may record what modification was made, but the meeting minutes must hold the definitive resolution (e.g., exact motion).
- Accepted-Duplicate: The comment was accepted without debate because it was a duplicate of some other comment.
- Rejected: The comment was rejected by the group
- Defer to Editor: The group deferred to the editor to resolve the comment (e.g., concluded that it was an editorial issue)
- Defer to Group: The group concluded to allow a sub-group (e.g., an ad-hoc group) to work off-line toward resolving the comment. The Notes field may indicate further information.

It would be useful to improve the comment database by adding these options to the "Resolution Status" field (these options may not be in the database accompanying this document):

- Tabled: Resolution of the comment was tabled until later. The notes field may indicate further information.
- Withdrawn: The commentor withdrew the comment.

4.3.2 New Document View

An editor also displays the document under revision on the overhead screen. With one projector, the editor can switch between this view and other views (e.g., database, and original document view).

Alternatively, with another projector (and another computer), both the comment view and the document view can be displayed simultaneously, thus saving meeting time switching back and forth between the two views. The main benefit here is for the group to simultaneously see both the database "reason" field and the comment's in-context document view, and save time switching back and forth. Also, the database view projector can be used to project other related documents to which the committee might need to refer, such as prior revisions, contributions, etc.

The document under revision contains change bars and red-lines for each comment to be resolved. As comments are debated, altered, and resolved, the editor performs the appropriate edits.

With MS-Word, the key operation here is to perform functions with MS-Word's change bar features. If you click on, or highlight a change, a right-click brings up a menu that presents an option to accept or reject a change. Using the change bars, and the change accept/reject feature, group can clearly see how the document is changed.

With FrameMaker, a similar mechanism using tags, can be used. Please refer to section 4.1.5.3.

4.3.3 Meeting Order

802.16 has used the following procedures during the meeting to facilitate discussion, debate and resolution. Suggestions for improvement are welcome.

Overall, the chair senses consensus in the group and may ask that some comments be accepted unless an objection is raised. Each comment needing some debate requires a "champion" from the group, likely the commentor, who raises a motion to support the comment. Note that this motion may alter the comment . Then following standard parliamentary procedures, debate continues on the comment and is resolved by a vote. A useful exception to standard parliamentary procedures is the concept of a "friendly amendment," which is a shortcut of the "motion to amend" procedure. The friendly amendment feature allows the motion to be altered, subject to the motioner's and seconder's approval.

Comment resolution is recorded in two places: the database and the minutes. The meeting minutes must contain the definitive resolution of comments, including the basis of comment alteration. The minutes refer to the comment database, by indicating the comment number. The comment database serves as supplementary information, and a comment resolution database report can be appended to the minutes if necessary.

Potential improvement note: This process can be improved by somehow integrating the comment database with the minutes with a tool that perhaps could automatically generate minutes by recording motions, results, etc. along with the details of comment resolution. With such an integrated tool, the secretary could "drive" the comment database and minutes simultaneously, while an editor "drives" only the current document edits. Then, following the meeting a unified report can be generated that includes comment resolution details and meeting minutes.

4.3.4 Backup

Periodically, throughout the meeting, the editor(s) should make backup copies of both the comment database and the document being edited.

4.3.5 After Meeting

An editor publishes a comment resolution report (from the database). This can be included in the minutes if desired. Also, subject to editorial "clean-ups," the editor can publish a new revision of the document in question.

Ideally, at the close of the meeting, the editor(s) have:

- A new version of the document that is very close to being publishable
- A report of how all comments were resolved

5 Tools

This section itemizes the tools necessary for managing the process described in this document.

Microsoft Word. The procedures used thus far by 802.16 utilize MS-Word. The reasons for this are wide availability and the flexible change bars management features. Note that editors in other 802 groups use FrameMaker. As described in section 4.1.4, we (802.16) haven't identified change resolution features in FrameMaker that are as effective as in MS-Word.

Perl. Perl is a widely used programming language for developing scripts (programs), and has feature that simplify programs that manipulate text. Perl scripts used by 802.16 should be found in files accompanying this document. Perl is available for virtually all computing platforms. The author uses Perl on various UNIXes and for Windows, uses Perl in the "cygwin" package, which is a UNIX command shell emulation environment for Windows platforms. It is a free package, found at: http://www.cygnus.com/cygwin.

Microsoft Access. All comments received, and their resolution status, are recorded in an MS-Access database. An empty database (a database template), that contains comment field definitions, on-screen forms and report formats should be found in a file accompanying this document.

Adobe Acrobat Exchange. Used to generate and publish and manipulate Portable Document Format (PDF) files. The primary use for this is that the MS-Access reports are exported only to printer formats and PDF files. And, since published documents require certain front matter (e.g., 802.16 submission template), Exchange is necessary to prepend front matter to MS-Access reports.

Email. Common Internet email tools are used to submit and receive comments.

A. Example call for comments

```
*
*  802.16.3 TASK GROUP SEEKS SPECIFIC COMMENTS ON FUNCTIONAL REQUIREMENTS
*
* Call for Comments on the Functional Requirements Working Draft
*

The 802.16.3 Task Group is working on a document that describes functional
requirements that directly affect the development of the 802.16.3 air
interface standard for broadband wireless access systems, as described in the
Project Authorization Request (http://ieee802.org/16/sub11/par).  It seeks
input and review by 802.16 members, participants and the industry to ensure
that the 802.16 standards meet industry expectations.

You may recall a similar effort 802.16 undertook to develop functional
requirements for the 802.16.1 air interface.  We intend to use the same
procedures in 802.16.3 as was done in 802.16.1 for comment resolution.
The exception is that, at least for this first batch of comments, we will use
a simple majority rule to resolve each comment, even "technical" ones (as we
decided at session #6).  As in 802.16.1, all comments will be incorporated
into a database and two documents will be produced prior to session #7: A
database "dump" that is sorted by page #/line #, and a new version of the
functional requirements document, with each comment edited-in and noted by
change bars.  Then, at session #7, the 802.16.3 task group will proceed by
considering each and every comment with discussion and vote.  Comments marked
"editorial" in nature may be lumped together in one motion and voted-in
together.

The current working draft is based on the 802.16.1 functional requirements
document.  To create the first working document, George Fishel made changes to
the 802.16.1 requirements that, in his view, target the requirements for BWA
under 11 GHz.  At session #6, the 802.16.3 task group decided to use this
document as a base, and undertake a formal comment/resolution process going
forward.  As the first step in the process, this message solicits comments
from industry to turn George's contribution into a "consensus" requirements
document.

The current working document can be found at:
http://ieee802.org/16/sub11/docs/802163-00_02r0.pdf

For those familiar with the 802.16.1 functional requirements, the editor
created a document (using the MS-Word "compare documents" feature) that
redlines the differences between the 802.16.3 functional requirements and
802.16.1 functional requirements. This "diffs" document can be found at:
http://ieee802.org/16/sub11/contrib/802163c-00_00.pdf.

The current working draft is open for comment regarding only specific changes:
insertion, deletion or change of any part of the document.  Below are detailed
instructions for submitting comments.  The instructions MUST be followed or
comments will be respectfully rejected by the editor and will not be entered
into the comment database.  For a rejected comment to be considered, the
commentor receiving a rejection MUST resubmit the comment(s) using the
instructions found below:

***Each comment on the document MUST use the comment submittal form found at:
 http://ieee802.org/16/sub11/commentform.txt.  The form is
 ASCII text-based (DOS text with CR/LF at the end of each line), and
 accommodates one "edit request."  It is filled-out with a short example.

***Commentors MUST replace the example text with their own information.
```

***Commentors MUST NOT delete or change the delimiters which are surrounded by square brackets ([]), as we need these to automatically import the submitted comment forms into a database.

***Commentors SHOULD wrap/fold each line at 80 columns or less. The only exception is for a table or other format that can not be rendered in 80 columns).

***Commentors MUST fill out all of the fields, which are:
- -Last Name
- -First Name
- -Starting page number
- -Starting line number
- -"T" or "E": Technical (content-related) or Editorial (typos, grammar, etc.)
- -A detailed description of the proposed edit
- -A reason for the edit

***The commentor MUST send the comments directly to the email reflector (mailto:stds-802-16@ieee.org). This allows other participants to see, and prepare for, all of the comments and avoid overlapping comments.

***The prefix "SUB11 COMMENT:" MUST appear on the subject line of the message. This allows participants to easily recognize COMMENT in their in-box, and non-participants to conveniently skip-over COMMENT messages.

***Each email message MUST NOT contain attachments such as MS-Word files, text files, Visio files, etc.

***Each email message MUST contain at least one comment form in the body of the message.

***An email message MAY contain more than one filled-out comment form in the body of a message.

***If multiple comment forms are sent in one message, each comment form MUST contain all of the fields filled out as required (see above), including the commentor's name.

***Comments regarding graphics, figures, drawings and tables MUST be fully described in text form. If such graphic-oriented comments can not be rendered or described in text form, the commentor MAY contact the editor for "special" consideration. We will accommodate these requests by associating a graphical figure with the text comment form.

***Commentors SHOULD send each comment message as a "reply" to the editor's "request for comments" message so that people who browse the reflector archive on our web page can more conveniently skip over the sysreq comments. Note that you may have to override the subject line with the "SYSREQ COMMENT:" prefix (see above).

The editor will gather all of the proposed changes (comments) and integrate them into the document and publish it as a new draft. Overlapping and conflicting comments will be clearly recognizable in the new draft.

Submissions will be considered non-confidential and will be posted for public access on the 802.16.3 area of the 802.16 Web Site.

For consideration for incorporating changes in the next working draft of the System Requirements Document, valid comments that follow these instructions MUST be received no later than close of business, Thursday April 27, 2000.

Thank you,

George Fishel
Communications Consulting Services
Shermans Dale, PA
717-582-2507
grfishel@pa.net

Brian Petry
3Com Corp.
San Diego, CA
858-674-8533
brian_petry@3com.com

B. Comment Form

[Submitter's Last Name]
Petry

[Submitter's First Name]
Brian

[Starting Page #]
9

[Starting Line #]
19

[(T)echnical for Content-Related Material; (E)ditorial for typos, grammar, etc.]
E

[Detailed Description of Proposed Insertion, Deletion, Change]
Insert "not" before "to provide error correction"

[Reason for Edit]
To fix a typo in drafts #1 and #2

C. On-line resolution instructions

*
* 802.16 SYSTEM REQUIREMENTS GROUP ON-LINE COMMENT RESOLUTION ANNOUNCEMENT
*

The 802.16 System Requirements Task Group seeks to resolve comments on draft 4 of their document. We received 144 comments that we hope to resolve through an on-line voting process described below. The objectives of on-line comment resolution are to save time at face-to-face meetings and accelerate the standardization process. Your participation is necessary to make it successful.

After this phase of on-line voting is complete, we will publish another version (draft 5) of the document which will be available for another round of comments prior to the 802.16 November 8-11 session. Hopefully, that round of comments will be the final one, and at the November session will resolve those comments and "finalize" the document.

To achieve consensus on the document, we need your participation now, during this on-line resolution process. 144 comments to resolve may seem like a daunting task, but hopefully this process will be as simple and painless as reasonably possible. 53 of the comments were marked "editorial" in nature and should be simple to resolve.

Here are the instructions, and rules, for participation in the voting process.

You must be a member of the 802.16 working group and have voting privileges (your name must appear on this list: http://grouper.ieee.org/groups/802/16/members.html).

You must submit a ballot by submitting a simple text form which you fill-out. The form is at http://grouper.ieee.org/groups/802/16/sysreq/ballot1.txt. You must submit the form via email to the 802.16 sysreq editor (mailto:brian_petry@3com.com).

Note that each comment has one of 4 possible votes:

(A) Accept
(M) Accept, but only if modified in some way
(B) Abstain (not qualified to vote or do not care)
(R) Reject

For an (M) or (R) vote, you must supply a reason for the rejection or required modification. We may have a second round of on-line resolution, prior to draft 5, to resolve (M) votes.

Note that the default vote is (A), which barring conflicts and overlaps with other votes, causes a comment to be accepted into the document. In other words, you must change the (A) on the form to vote otherwise.

You must vote on every comment.

You must fill out your name on the form.

If a ballot form is not filled-out properly, it will be rejected by the editor. If time permits prior to the vote deadline, you may resubmit a corrected form.

The deadline for submitting a valid ballot form is Wednesday, October 6, 12:00 PM Pacific Daylight Time.

The votes will be tallied by the editor and results published soon after the deadline. For comments marked editorial, a simple majority is required to pass. If a comment is marked technical, a 75% majority is required. If a comment does not receive a majority in any of the 4 possible votes, it is not accepted into the document. Some of these comments may be contentious, yet important to resolve. The commentor may facilitate on-line debate of these contentious comments using the 802.16 email reflector (mailto:stds-802-16@ieee.org) as an opportunity to reword and make compromises prior to the next round of comments.

The comment database is in a form that, hopefully, is convenient for members to review. It can be found at http://grouper.ieee.org/groups/802/16/sysreq/contributions/80216sc-99_33.pdf. A convenient way to fill out the ballot form is to have hard copies of draft 4 and the comment database while filling out the text form with a text editor. Here are the links, for your convenience:

http://grouper.ieee.org/groups/802/16/sysreq/contributions/80216s0-99_4.pdf

http://grouper.ieee.org/groups/802/16/sysreq/contributions/80216sc-99_33.pdf
http://grouper.ieee.org/groups/802/16/sysreq/ballot1.txt

George Fishel (Chair)
Communications Consulting Services
Shermans Dale, PA
717-582-2507
grfishel@pa.net

David Jarrett (Vice Chair)
Lucent Technologies
Milpitas, CA
408-952-7452
djarrett@lucent.com

Brian Petry (Editor)
3Com Corp.
San Diego, CA
858-674-8533
brian_petry@3com.com

D. On-line comment resolution form: Example

802.16 System Requirements Task Group Comment ballot form. 16 Sept., 1999.

You must be a member of the 802.16 working group and have voting privileges (your name must appear on this list: http://grouper.ieee.org/groups/802/16/members.html).

You must submit a ballot by submitting this form (http://grouper.ieee.org/groups/802/16/sysreq/ballot1.txt). You must submit the form via email to the 802.16 sysreq editor (mailto:brian_petry@3com.com).

Each comment has one of 4 possible votes:
(A) Accept
(M) Accept, but only if modified in some way
(B) Abstain (not qualified to vote or do not care)
(R) Reject

For an (M) or (R) vote, you must supply a reason for the rejection or required modification. We may have a second round of on-line resolution, prior to draft 5, to resolve (M) votes. Note that the default vote is (A), which barring conflicts and overlaps with other votes, causes a comment to be accepted into the document. In other words, you must change the (A) on the form to vote otherwise.

You must vote on every comment.

If a ballot form is not filled-out properly, it will be rejected by the editor. If time permits prior to the vote deadline, you may resubmit a corrected form.

The deadline for submitting a valid ballot form is Wednesday, October 6, 12:00 PM Pacific Standard Time.

You should attempt to fit your reason field on one line. But if you need more room, you must wrap/fold each line at 80 columns without typing a digit in the first column.

```
Your Last Name:
Your First Name:

Cmnt  Vote  Reason
----  ---   -----------------------------------------------------------------
114   (A)
115   (A)
116   (A)
117   (A)
118   (A)
119   (A)
```

…Etc. (truncated)

E. Example importable file

```
@Number@|@Last Name@|@First Name@|@Page Number@|@Line Number@|@Description of
Edit@|@Reason for Edit@|@Date Received@|@Resolution@|@Date Resolved@|@Comment
Type@|@Notes@|@Document@
1|@Abu-Dayya@|@Adnan@|4|11|@Replace "from 1 GHz to 10 GHz" with "2 GHz to 11 GHz"
**Editor's note: Rejected because change is already there**

@|@To be consistent with the PAR.

@|04/28/2000 0:00:00|@unresolved@||@Technical@||@802.16.3f-00/01@
2|@Abu-Dayya@|@Adnan@|4|17|@Replace "NLOS blockage" with "channel characteristics"

@|@A more meaningful metric in link design.

@|04/28/2000 0:00:00|@unresolved@||@Technical@||@802.16.3f-00/01@
3|@Abu-Dayya@|@Adnan@|10|20|@replace "will be a difficult problem for" with "will be a
requirement  for"

@|@The target markets "residential and small businesses" mandate a
non-line-of-sight requirement. Hence, the airlink should be robust in
multipath environments.


@|04/28/2000 0:00:00|@unresolved@||@Technical@||@802.16.3f-00/01@
4|@Abu-Dayya@|@Adnan@|10|21|@Delete "The 802.16.3 system capacity ...will also be
difficult"

@|@Not a very meaningful sentence; serves no purpose.

@|04/28/2000 0:00:00|@unresolved@||@Technical@||@802.16.3f-00/01@
5|@Freedman@|@Avi@|1|4|@Change "Broadband Wireless Access (BWA)"  to "Wideband Wireless
Access
(WWA)"  all over the document.

@|@To distinguish 802.16.3 activity from 802.16.1

@|04/28/2000 0:00:00|@unresolved@||@Technical@||@802.16.3f-00/01@
6|@Freedman@|@Avi@|1|45|@Delete comma after "networks"

@|@Grammar

@|04/28/2000 0:00:00|@unresolved@||@Editorial@||@802.16.3f-00/01@
7|@Freedman@|@Avi@|3|29|@Exchange the paragraph between lines 29 and 35 (starting with "A
broadband
wireless access" with the next paragraph (between lines 37 and 42, starting
with "The target markets to be addressed...)

@|@The definition of the markets is more important than the examples of
```

```
possible services.

@|04/28/2000 0:00:00|@unresolved@||@Technical@||@802.16.3f-00/01@
8|@Freedman@|@Avi@|3|30|@Change "markets" to "services"

@|@The technologies provide services and are not special to markets.

@|04/28/2000 0:00:00|@unresolved@||@Technical@||@802.16.3f-00/01@
9|@Freedman@|@Avi@|4|7|@Insert space between "802.16.3" and "MAC"

@|@Typo


@|04/28/2000 0:00:00|@unresolved@||@Editorial@||@802.16.3f-00/01@
10|@Freedman@|@Avi@|4|11|@Insert space between "to" and "3.5"

@|@Typo


@|04/28/2000 0:00:00|@unresolved@||@Editorial@||@802.16.3f-00/01@
11|@Freedman@|@Avi@|6|1|@Change "A base" to "Base"

@|@Refer to "interfaces"

@|04/28/2000 0:00:00|@unresolved@||@Editorial@||@802.16.3f-00/01@
12|@Freedman@|@Avi@|6|11|@Remove space in "comm on"

@|@Typo


@|04/28/2000 0:00:00|@unresolved@||@Editorial@||@802.16.3f-00/01@
13|@Freedman@|@Avi@|6|12|@Add optional Central Control Station and more base station to
diagram

@|@The diagram may be adequate for 802.16.1 but it is too simplistic for
802.16.3, where there is hihg influence between stations. A diagram similar
to the ETSI TM4 refernce diagram should be used.

@|04/28/2000 0:00:00|@unresolved@||@Technical@||@802.16.3f-00/01@
14|@Freedman@|@Avi@|7|17|@Add "mainly" between "be" and "packet".

@|@There is no reason to assume nor to limit voice services to Voice over
packets.  Other mechanisms may be found to be better.
```

Etc. (truncated)

F. Perl scripts

These are some perl scripts that help automate the manipulation of text-based comments and on-line ballots received via email. Perl is available on all computing platforms. For Windows, the author uses Perl from the cygwin package, which is a free UNIX command shell and tool environment, found at http://www.cygnus.com/cygwin.

F.1. unix2dos.pl

This script converts from UNIX (LF at the end of a line) to DOS text (CR/LF at the end of each line) format.

```
#!/usr/local/bin/perl

#
# Usage: cat whatever | unix2dos.pl > outputfile
#
# Adds a CR character to each LF if CR is not already there
#
while (<STDIN>) {
      if (substr($_, -2, 1) eq "\r") {
            print $_;
      } else {
            print substr($_, 0, -1), "\r\n";
      }
}
```

F.2. dos2unix.pl

```
#!/usr/local/bin/perl

#
# Usage: cat whatever | dos2unix.pl > outputfile
#
# deletes a trailing CR character from each line in the source file
#
while (<STDIN>) {
      if (substr($_, -2, 1) eq "\r") {
            print substr($_, 0, -2) . "\n";
      } else {
            print $_;
      }
}
```

F.3. fix.pl

```
#!/usr/local/bin/perl

#
# Fixes some syntax errors that commentors may have made
# outputs to stdout
#
# usage:
# fix.pl comments.txt

$progname = "fix.pl";
$argc = scalar(@ARGV);

if ($argc != 1) {
      printf("Usage: $progname commentfile.txt\n");
      exit(1);
}

open(cfile, @ARGV[0]) or die "Can't open stack @ARGV[0]: $!\n";

$lastline = "";

while (<cfile>) {
      if (/^\[/) {
            if ($lastline ne "") {
                  print "\n";
            }
            $lastline = $_;
      }
      print $_;
}
```

F.4. cimport.pl

```
#!/usr/local/bin/perl

#
# Converts human-readable comments to a form importable by MS-Access
# outputs to stdout
#
# usage:
# cimport.pl comments.txt startnum date

$progname = "cimport.pl";
$argc = scalar(@ARGV);

if ($argc != 3) {
      printf("Usage: $progname commentfile.txt startnum date\n");
      exit(1);
}

open(cfile, @ARGV[0]) or die "Can't open stack file @ARGV[0]: $!\n";

# states:
# 0: none
# 1: Last Name
# 2: First Name
# 3: Start Page
# 4: Start Line
# 5: T/E
# 6: Description
# 7: Reason

$nc = 0; #cummulative number of comments
$state = 0;
$ci = @ARGV[1]; #comment index
$date = @ARGV[2];
#@descr[0] = "nothing";
#@reason[0] = "nothing";

#print the field headings
print "\@Number\@|\@Last Name\@|\@First Name\@|\@Page Number\@|\@Line
Number\@|\@Description of Edit\@|\@Reason for Edit\@|\@Date
Received\@|\@Resolution\@|\@Date Resolved\@|\@Comment Type\@|\@Notes\@|\@Document\@\n";

# print a comment record
sub printc {
      print "$ci|\@$last\@|\@$first\@|$page|$line|\@";
      foreach $line (@descr) { print $line; }
      print "\@|\@";
      foreach $line (@reason) { print $line; }
      print "\@|";
      print "$date 0:00:00|\@unresolved\@||\@$type\@||\@802.16.3f-00/01\@\n";
}

while (<cfile>) {
      if (/^\[.*Last/) {
            if ($state == 7) {
                  $#descr--;  #remove extra line
                  $#reason--; #remove extra line
                  printc();
                  $ci++;
            }
            $state = 1; $nc++; next;
      }

      if (/^\[.*First/) { $state = 2; next; }

      if (/^\[Starting Page/) { $state = 3; next; }

      if (/^\[Starting Line/) { $state = 4; next; }

      if (/^\[\(T/) { $state = 5; $_ = <cfile>; next; } # eat an extra line

      if (/^\[Detailed/) { $state = 6; next; }

      if (/^\[Reason/) { $state = 7; next; }

      if ($state == 1) { chomp($last = $_); $state = 0; $#descr = 0; $#reason = 0; next; }

      if ($state == 2) { chomp($first = $_); $state = 0; next; }

      if ($state == 3) {
            chomp($page = $_);

            if (!($page =~ /^[0-9]*$/)) {
                  print STDERR "Line ", $. ,": Invalid page number\n";
            }
            $state = 0;
            next;
      }

      if ($state == 4) {
            chomp($line = $_);

            if (!($line =~ /^[0-9]*$/)) {
                  print STDERR "Line ", $. ,": Invalid line number\n";
            }
            $state = 0;
            next;
      }
      if ($state == 5) {
            if (/T/) { $type = "Technical"; }
                  else { $type = "Editorial"; }
            $state = 0;
            next;
      }

      if ($state == 6) {
            $x = $#descr;
            @descr[$x+1] = $_;
            next;
      }
      if ($state == 7) {
            $x = $#reason;
            @reason[$x+1] = $_;
            next;
      }
}

printc();

print STDERR "$nc comments processed; last index=$ci\n";
```

F.5. votecount.pl

```
#!/usr/local/bin/perl

#
# Counts votes from text files
#
# usage:
# votecount.pl file1 file2 ...

$progname = "votecount.pl";
$argc = scalar(@ARGV);

$startitem = 114;
$maxvotes = 144;
$voters=0;

# For each file specified on the command line
foreach $vfilename (@ARGV) {

      $item = $startitem;     #starting item number
      $votecount = 0;
      $voters++;

      #print "$vfilename\n";

      open(vfile, $vfilename) or die "Can't open file $vfilename: $!\n";

      while (<vfile>) {
            if (/^$item/) {
                  @fields = split;
                  #print "$vfilename @fields[1]\n";

                  #if ($item == $startitem) { print "X$fields[1]X\n"; }

                  if ($fields[1] =~ /\(A\)/) {
                        @accept[$item]++;
                  }

                  if ($fields[1] =~ /\(R\)/) {
                        @reject[$item]++;
                  }

                  if ($fields[1] =~ /\(M\)/) {
                        @modify[$item]++;
                  }
```

```
                        if ($fields[1] =~ /\(B\)/) {
                              @abstain[$item]++;
                        }

                        $item++;
                        $votecount++;
                  }
            }

            if ($votecount != $maxvotes) {
                  print "Error: file $vfilename has invalid vote count; counted $votecount\n";
            }

            close(vfile);
}

$techvotes = int $voters * .75;

open(ctfile, "commenttypes.txt") or die "Can't open file commenttypes.txt: $!\n";

while (<ctfile>) {
      @fields = split;
      if (/echnical/) {
            @technical[@fields[0]] = 1;
      } else {
            @technical[@fields[0]] = 0;
      }

      #printf "%d: %d\n", @fields[0], @technical[@fields[0]];
}

close(ctfile);


print "Item# Type Acc  Rej Mod aBs Result\n";
$item = $startitem;
while ($item < $startitem + $maxvotes) {
      $result[$item] = "F";

      if (@technical[$item]) {
            $t = "T";
      } else {
            $t = "E";
      }
      printf "%3d: %4s %3d %3d %3d %3d", $item, $t, @accept[$item], @reject[$item],
@modify[$item], @abstain[$item];

      if (@accept[$item] +
            @reject[$item] +
            @modify[$item] +
            @abstain[$item] != $voters) {

            print "***Vote miscount on item $item\n";
            $item++;
            next;
      }


      #if accept wins
      if (@accept[$item] > @reject[$item] &&
             @accept[$item] > @modify[$item] &&
             @accept[$item] > @abstain[$item]) {
```

```
            $resultstr = "   Accepted by ";
            if (@accept[$item] >= $techvotes) {
                  $resultstr .= ">= 75\% majority.";
                  $result[$item] = "A";
            } else {
                  if (@technical[$item]) {
                        $margin = $techvotes - @accept[$item];
                        $resultstr = "   Failed to achieve 75\% accept votes by $margin.";
                        $result[$item] = "FA";
                  } elsif (@accept[$item] > int (.50 * $voters)) {
                        $resultstr .= ">= 50\% majority.";
                        $result[$item] = "A";
                  } else {
                        $resultstr .= "simple majority.";
                        $result[$item] = "A";
                  }
            }

            print "$resultstr\n";
      }

      #if reject wins
      elsif (@reject[$item] > @accept[$item] &&
             @reject[$item] > @modify[$item] &&
             @reject[$item] > @abstain[$item]) {

            $resultstr = "   Rejected by ";
            if (@reject[$item] >= $techvotes) {
                  $resultstr .= ">= 75\% majority.";
                  $result[$item] = "R";
            } else {
                  if (@reject[$item] > int (.50 * $voters)) {
                        $resultstr .= ">= 50\% majority.";
                        $result[$item] = "R";
                  } else {
                        $resultstr .= "simple majority.";
                        $result[$item] = "R";
                  }
            }

            print "$resultstr\n";
      }

      #if modify wins
      elsif (@modify[$item] > @reject[$item] &&
             @modify[$item] > @accept[$item] &&
             @modify[$item] > @abstain[$item]) {

            $resultstr =  "   Accepted, but only if modified by ";
            if (@modify[$item] >= $techvotes) {
                  $resultstr .= ">= 75\% majority.";
                  $result[$item] = "M";
            } else {
                  if (@modify[$item] > int (.50 * $voters)) {
                        $resultstr .= ">= 50\% majority.";
                        $result[$item] = "M";
                  } else {
                        $resultstr .= "simple majority.";
                        $result[$item] = "M";
                  }
            }
            print "$resultstr\n";
      }

      #if abstain wins
      elsif (@abstain[$item] > @reject[$item] &&
             @abstain[$item] > @modify[$item] &&
             @abstain[$item] > @accept[$item]) {

            $resultstr = "   Abstains win by ";
            if (@abstain[$item] >= $techvotes) {
                  $resultstr .= ">= 75\% majority.";
                  $result[$item] = "B";
            } else {
                  if (@abstain[$item] > int (.50 * $voters)) {
                        $resultstr .= ">= 50\% majority.";
                        $result[$item] = "B";
                  } else {
                        $resultstr .= "simple majority.";
                        $result[$item] = "B";
                  }
            }

            print "$resultstr\n";
      }

      else {
            if (@technical[$item]) {
                  print "   Failed to achieve majority (tie).\n";
            } else {
                  print "   *** Tie vote ***.\n";
            }
      }

      $item ++;
}

#
# Read commentor names
#
open(nfile, "number_name.txt") or die "Can't open file commenttypes.txt: $!\n";

while (<nfile>) {
      @fields = split;
      $x = substr($_, index($_, @fields[1]));
      @cname[@fields[0]] = $x;
      #print "$x";
}

close(nfile);

#
# Extract reasons from failed comments.  Indicate voter name and vote
#

#
# First, find all comments that might have passed, but failed to get 75%
# majority and print the M votes.
#
print "\n***Reason fields for comments that failed to achieve 75% majority
follow:***\n\n";
```

```
foreach $vfilename (@ARGV) {

      $item = $startitem;     #starting item number

      open(vfile, $vfilename) or die "Can't open file $vfilename: $!\n";

      $state = 0;

      while (<vfile>) {
            @fields = split;

            if (/Last/) {
                  $lastname = @fields[3];
                  next;
            }

            if (/First/) {
                  $firstname = @fields[3];
                  next;
            }

            if (/^$item/) {

                  $state = 0;

                  # if this item failed to achieve a majority
                  if ($result[$item] eq "FA") {
                        $v = substr $fields[1], 0, 3;

                        if ($v eq "(M)") {
                              $state = 1;
                              $reason = "   $firstname $lastname voted $v because:\n";

                              $line = substr($_, (index $_, @fields[1]) + 3);
                              $line =~ s/^\s*//; #strip leading white space
                              $reason .= "      $line";

                              @reasons[$item] .= $reason;
                        }
                  }

                  $item++;
                  next;
            }

            #if processing a multiline description

            if ($state == 1) {
                  s/^\s*//; #strip leading white space

                  # if it's a blank line, don't save it
                  if ($_ ne "") {
                        @reasons[$item-1] .= "      $_";
                  }
            }
      }
}

#
# Next, print the R votes.
#
foreach $vfilename (@ARGV) {
	$item = $startitem;	#starting item number

	open(vfile, $vfilename) or die "Can't open file $vfilename: $!\n";

	$state = 0;

	while (<vfile>) {
		@fields = split;

		if (/Last/) {
			$lastname = @fields[3];
			next;
		}

		if (/First/) {
			$firstname = @fields[3];
			next;
		}

		if (/^$item/) {

			$state = 0;

			# if this item failed to achieve a majority
			if ($result[$item] eq "FA") {
				$v = substr $fields[1], 0, 3;

				if ($v eq "(R)") {
					$state = 1;
					$reason = "   $firstname $lastname voted $v because:\n";

					$line = substr($_, (index $_, @fields[1]) + 3);
					$line =~ s/^\s*//; #strip leading white space
					$reason .= "	$line";

					@reasons[$item] .= $reason;
				}
			}

			$item++;
			next;
		}

		#if processing a multiline description

		if ($state == 1) {
			s/^\s*//; #strip leading white space

			# if it's a blank line, don't save it
			if ($_ ne "") {
				@reasons[$item-1] .= "	$_";
			}
		}
	}
}

#
# Next, print the B votes.
#
foreach $vfilename (@ARGV) {
      $item = $startitem;     #starting item number

      open(vfile, $vfilename) or die "Can't open file $vfilename: $!\n";

      $state = 0;

      while (<vfile>) {
            @fields = split;

            if (/Last/) {
                  $lastname = @fields[3];
                  next;
            }

            if (/First/) {
                  $firstname = @fields[3];
                  next;
            }

            if (/^$item/) {

                  $state = 0;

                  # if this item failed to achieve a majority
                  if ($result[$item] eq "FA") {
                        $v = substr $fields[1], 0, 3;

                        if ($v eq "(B)") {
                              $reason = "   $firstname $lastname voted $v because:\n";

                              $line = substr($_, (index $_, @fields[1]) + 3);
                              $line =~ s/^\s*//; #strip leading white space


                              # check for empty reason
                              if ($line ne "") {
                                    $state = 1;
                                    $reason .= "      $line";
                              } else {
                                    $reason .= "      No reason given.\n";
                              }

                              @reasons[$item] .= "$reason";
                        }
                  }

                  $item++;
                  next;
            }

            #if processing a multiline description

            if ($state == 1) {
                  s/^\s*//; #strip leading white space

                  # if it's a blank line, don't save it
                  if ($_ ne "") {
                        @reasons[$item-1] .= "      $_";
                  }
            }
      }
}
```

```
#foreach $reason (@reasons) {
#     print $reason;
#}

$item = $startitem;
while ($item < $startitem + $maxvotes) {
      if ($result[$item] eq "FA") {
            print "$item: author=@cname[$item]";
            print @reasons[$item];
      }
      $item++;
}

# clear out the reasons buffer
@reasons = 0;

#
# Now, display the reason fields for any vote
#
print "\n***Reason fields for all comments follow:***\n\n";
foreach $vfilename (@ARGV) {

      $item = $startitem;     #starting item number

      open(vfile, $vfilename) or die "Can't open file $vfilename: $!\n";

      $state = 0;

      while (<vfile>) {
            @fields = split;

            if (/Last/) {
                  $lastname = @fields[3];
                  next;
            }

            if (/First/) {
                  $firstname = @fields[3];
                  next;
            }

            if (/^$item/) {

                  $state = 0;

                  $v = substr $fields[1], 0, 3;

                  $reason = "   $firstname $lastname voted $v because:\n";

                  $line = substr($_, (index $_, @fields[1]) + 3);
                  $line =~ s/^\s*//; #strip leading white space
                  $reason .= "      $line";

                  # check for empty reason
                  if ($line ne "") {
                        $state = 1;
                        @reasons[$item] .= "$reason";
                  }

                  $item++;
                  next;
            }

            #if processing a multiline description

            if ($state == 1) {
                  s/^\s*//; #strip leading white space

                  # if it's a blank line, don't save it
                  if ($_ ne "") {
                        @reasons[$item-1] .= "      $_";
                  }
            }
      }
}

#foreach $reason (@reasons) {
#     print $reason;
#}

$item = $startitem;
while ($item < $startitem + $maxvotes) {
      print "$item: author=@cname[$item]";
      print @reasons[$item];
      $item++;
}
```

G. Database Specification

This appendix strives to document the database design: fields, forms, reports, etc.

G.1. Comment Table Fields

Number. The comment number. This number is typically assigned automatically during import, and automatically assigned if comments are entered manually. This the only “key” defined for the database. Each comment must have a unique number.

Last Name. The commentor’s last name.

First Name. The commentor’s first name.

Page Number: The starting page number of the change in question

Line Number: The starting line number (within the page) of the change in question.

Description of Edit: A detailed description of the edit: add/delete/change

Reason for Edit: A descriptive reason for requesting the change.

Date Received: The date a comment was received, or imported into the database.

Resolution: The resolution status. This field is enumerated into various values. It records the current status of the comment.

Date Resolved. The date the comment was resolved.

Comment Type. The type of comment: Editorial (absolutely needs no debate) or Technical (might need some debate).

Notes. Miscellaneous notes that might be useful to record, typically any special resolution that happened resulting from debate.

Document. A reference to the document in question (e.g., the document number).

G.2. Forms

Comment Entry. Used to enter comments manually if the automatic import functions fail for some reason, or a small number of comments don’t warrant the “import” process. The [Tab] key can be used to cycle through fields in a reasonable sequence.

Comment Resolution. Used off-line (not in a large meeting setting) to enter comment resolutions. The [Tab] key cycles through fields in an appropriate manner for this. This form is not used in a large meeting setting, but by the editor to manually enter resolutions (e.g., results from email votes).

Comment Resolution Big Screen. This is usually the only form used in a meeting setting during comment resolution and debate.

G.3. Reports

There are various reports defined in the comment database file, and they are very simple to set up or modify.

G.4. Potential Improvements

It should be possible to automate the comment submittal process using a web page.

Possible improvements for new database fields and changes:

- Needs confirmation: Boolean. This new field could be used to hold comments resolved on-line, with their corresponding resolution status (e.g., accepted/modified/rejected/etc.), in a “limbo” state until the group confirms their resolution at the next meeting.
- Macros. If someone is ambitious to write Access Visual Basic Macros (this author isn’t), the database operations might be enhanced so that meeting face-to-face time could be conserved. Here are some ideas:
  - Sort the record list viewed by a form in a hierarchy (e.g., page#/line#)
  - Automatically switch to the original MS-Word document and locate the page # and line #.
  - Implement keystroke shortcuts to zap fields (like the resolution status) quickly, or set the cursor in a particular field. Basically, these are mouse-motion-time optimizations.
  - Automatically insert today’s date into the “Date Resolved” field.