

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NINTENDO CO., LTD., and
NINTENDO OF AMERICA INC.,
Petitioners

v.

MALIKIE INNOVATIONS LTD.,
Patent Owner

Case IPR2026-00007
US Patent No. 9,542,571 B2

DECLARATION OF INGRID HSIEH-YEE, PH.D.

TABLE OF CONTENTS

	Page
I. INTRODUCTION AND QUALIFICATIONS	3
A. Qualifications and Professional Experience.....	3
II. BACKGROUND OF LIBRARY CATALOGING	4
A. Machine-Readable Cataloging (“MARC”)	4
B. Library Online Catalogs	7
C. Scope of this Declaration	10
D. Evidence Considered in Forming My Opinions.....	12
III. AUTHENTICITY AND PUBLIC AVAILABILITY OF <i>WINDOWS</i> <i>GUIDELINES BOOK</i> (EX. 0001)	12
A. Authentication	12
B. Records of the Library of Congress	15
C. Copyright Registration Record.....	20
D. Records of the University of Michigan Libraries	20
E. Internet Archive Record	23
F. Usage Records.....	24
G. Summary of My Opinion on the <i>Windows Guidelines Book</i>	24
IV. AUTHENTICITY AND PUBLIC AVAILABILITY OF <i>GOMBERG</i> (EX. 0002)	26
A. Authentication	26
B. Records of the Penn State University Libraries	28
C. Records of the University of Minnesota Libraries.....	31
D. Internet Archive Records	34
E. Usage Records.....	37
F. Summary of My Opinion on <i>Gomberg</i>	37
V. CONCLUSION.....	39

I, Ingrid Hsieh-Yee, declare as follows:

I. Introduction and Qualifications

1. I am over the age of eighteen and otherwise competent to make this declaration.

2. I have been retained by counsel for Petitioner to provide my expert opinion in connection with the above-captioned proceeding (“IPR”) regarding the public availability of certain prior art references in relation to U.S. Patent No. 9,542,571 B2 (“Challenged Patent”).

3. I am being compensated for my time in connection with this IPR at my current consulting rate. I am also being reimbursed for any reasonable expenses associated with my work and testimony in this investigation. My compensation is not contingent on the results of my study, the substance of my opinions, or the outcome of this matter.

A. Qualifications and Professional Experience

4. My complete qualifications and professional experience are described in my academic curriculum vitae (**Appendix A**). The following is a brief summary of my relevant qualifications and professional experience.

5. I was a professor in the Department of Library and Information Science at the Catholic University of America for 32 years and was conferred the title of Professor Emerita in December 2022. I have experience working in an academic

library, a medical library, and a legislative library. I hold a Ph.D. and a Masters in Library and Information Studies from the University of Wisconsin-Madison.

6. I am an expert on library cataloging and classification and have published two editions on this subject, *Organizing Audiovisual and Electronic Resources for Access: A Cataloging Guide* (2000 and 2006). I taught a variety of courses, including Cataloging and Classification, Advanced Cataloging and Classification, Organization of Internet Resources, Organization of Information, Metadata, Digital Content Creation and Management, Internet Searches and Web Design, Information Literacy Instruction, Advanced Information Retrieval and Analysis Strategies, and The Information Professions in Society. I am familiar with metadata schema design and implementation. In my teaching, I covered the design and implementation of metadata in databases, search engines, digital repositories, digital libraries, and digital archives. I also covered how information organization affects the discovery and access to digital resources on the Internet. My research interests cover cataloging and classification, information organization, metadata, information retrieval, information architecture, digital collections, scholarly communication, social media, user interaction with information systems, and others

II. Background of Library Cataloging

A. Machine-Readable Cataloging (“MARC”)

7. I am very familiar with a library cataloging encoding standard known

as the “Machine-Readable Cataloging” standard, also known as “MARC,” which became the national standard for sharing bibliographic data in the United States by 1971 and the international standard by 1973. MARC is the primary communications protocol for the transfer and storage of bibliographic metadata in libraries. Experts in my field reasonably rely upon MARC records when forming their opinions.

8. A MARC record consists of several fields, each of which contains specific data about the work. Each field is identified by a standardized, unique, three-digit code corresponding to the type of data that follows. **Appendix B** is a true and correct copy of Parts VII to X of “Understanding MARC Bibliographic: Machine-Readable Cataloging” (<https://www.loc.gov/marc/umb/um07to10.html>), a brief description and tutorial published by the Library of Congress in 2009 that explains commonly used MARC fields. For example, the personal author of a work is recorded in Field 100, the title is recorded in Field 245, publisher information is recorded in Field 260, the physical volume and characteristics of a publication are recorded in Field 300, and topical subjects are recorded in Field 650. More details of MARC21 format for Bibliographic Data are available on the Library of Congress website at <https://www.loc.gov/marc/bibliographic/>.

9. I am very familiar with library cataloging and processing practices and the relationships between OCLC (Online Computer Library Center) and library catalogs, and how libraries create and share their records to help the public discover

and access information.

10. OCLC is the largest bibliographic network of the world, and its WorldCat database contains more than 559 million records from thousands of member institutions (many of which are libraries of some type) in more than 100 countries. OCLC was founded in 1967 to promote and support library cooperation. According to the third article of the “Amended Articles of Incorporation of OCLC [Online Computer Library Center], Inc.,” OCLC was created “to establish, maintain, and operate a computerized library network and to promote the evolution of library use, of libraries themselves, and of librarianship, and to provide processes and products for the benefit of library users and libraries, including such objectives as increasing availability of library resources to individual library patrons and reducing the rate of rise of library per-unit costs, all for the fundamental public purpose of furthering ease of access to and use of the ever-expanding body of worldwide scientific, literary, and educational knowledge and information” (source: <https://www.oclc.org/content/dam/oclc/membership/articles-of-incorporation.pdf>).

11. OCLC members can contribute original cataloging records in MARC to the system or derive cataloging records from existing records, an activity referred to as “copy cataloging.” When an OCLC participating institution acquires a work, it can create an original MARC record for the work in OCLC’s WorldShare system (a system for catalogers to create and share MARC records), and the system will

automatically generate a code for the date of record creation in the yymmdd format in the 008 field, and the creating library's OCLC symbol is recorded in subfield "a" of the 040 field. Once the MARC record is created in WorldShare, the record is stored in the OCLC WorldCat database and becomes available to other OCLC members for adoption to their local online catalogs (*i.e.*, copy cataloging). "WorldCat is the world's most comprehensive database of information about library collections" (source: WorldCat, <https://www.oclc.org/en/worldcat.html>).

12. After a MARC record is created in WorldShare, it also becomes searchable and viewable on WorldCat.org, which is a free web portal for users to explore more than 10,000 libraries worldwide. Records in WorldCat.org, however, are not presented in MARC fields. Instead, the data elements are labeled to help users interpret the records. Thus, the information stored in MARC records in WorldShare is available to the interested public through the user-friendly WorldCat web portal.

B. Library Online Catalogs

13. Library online catalogs gained acceptance in the early 1980s and many libraries migrated their systems to the World Wide Web in the mid-1990s. Library online catalogs are based on MARC records that represent their collections in order to help the public understand what materials are publicly accessible in those libraries. Most libraries with online catalogs have made their catalogs freely available on the Web. These online catalogs offer user-friendly search interfaces. Strong user interest

in keyword searches and the popularity of Google have led to the “googlization” of library search systems. As a result, many library catalogs nowadays provide a single search box for users to conduct keyword searches, with additional support for searches by author, title, subject terms, and other data elements such as ISBN (International Standard Book Number). Library catalogs these days also offer features for users to narrow their search results by language, year, format, and other elements. Many libraries display MARC records on their online catalogs with labels for the data elements to help the public interpret MARC records. Many libraries also offer the option to display MARC records in MARC fields.

14. Libraries create MARC records for works they acquire, including books, serials, motion pictures, and publications in other formats. Monograph cataloging is fairly common in libraries, and most libraries make a newly cataloged monograph available to the public soon after the cataloging work is completed, usually within a week. Libraries can create original cataloging records or use an existing record in the OCLC WorldCat database to create a copy cataloging record. As soon as the cataloging record is completed, it is added to the library’s online catalog for users. If the record is an original record, it is also entered into the OCLC system. If it is a copy cataloging record, the library’s holding symbol is attached to the existing original record in the OCLC system to facilitate searching and interlibrary loan.

15. I am personally familiar with many online catalogs, databases, and search engines. In preparing for this declaration, I used the following authoritative information systems to search for records:

- ACM Digital Library (<https://dl.acm.org/>)
- Google Scholar (<https://scholar.google.com/>)
- Internet Archive (<https://archive.org/>)
- Libraries.org, a directory of libraries throughout the world (<https://librarytechnology.org/libraries/>)
- Online catalog of the Library of Congress (<https://catalog.loc.gov/>)
- Online catalog of the Pennsylvania State University Libraries (<https://catalog.libraries.psu.edu/>)
- Online catalog of the University of Michigan Libraries (<https://search.lib.umich.edu/catalog/>)
- Online catalog of the University of Minnesota Libraries (<https://www.lib.umn.edu/>)
- Public Records System of the US Copyright Office (<https://publicrecords.copyright.gov/>)
- Semantic Scholar (<https://www.semanticscholar.org/>)
- WorldCat.org (<https://search.worldcat.org/>)

C. Scope of this Declaration

16. I have been asked to offer an opinion on the public availability date of the following documents:

- *The Windows Interface Guidelines for Software Design*. (1995). Microsoft Press, **Ex. 0001** (“*Windows Guidelines Book*”);
- Gomberg, M., Evard, R., & Stacey, C. (1998, August). A comparison of large-scale software installation methods on NT and UNIX, in *Proceedings of the Large Installation System Administration of Windows NT Conference, August 5-8, 1998, Seattle, Washington, USA*, pp. 37-48, **Ex. 0002** (“*Gomberg*”).

17. I am informed by counsel and understand that a given reference is “published” upon a satisfactory showing that such a document has been disseminated or otherwise made publicly available to the extent that persons interested and ordinarily skilled in the subject matter or art could locate it exercising reasonable diligence and obtain the document. I have also been informed by counsel and understand that materials available in a library constitute “printed publications” if they are catalogued and indexed according to general library practices and protocols that make the references available and accessible to members of the interested public.

18. I have been informed by counsel that the Challenged Patent claims December 12, 2002, as its earliest priority date. I have assessed public accessibility

of the documents referenced below as of this claimed priority date.

19. I have been informed by counsel that the subject matter of this proceeding generally relates to computer security and application management systems.

20. I have been informed by counsel that a “person of ordinary skill in the art at the time of the invention” (POSITA) is a hypothetical person who is presumed to be familiar with the relevant field and its literature at the time of the inventions. I have been informed by counsel that this hypothetical person is also a person of ordinary creativity, capable of understanding the scientific principles and literature applicable to the pertinent field.

21. I have been informed by counsel that a POSITA as of December 2002 would have possessed a Bachelor’s degree in software engineering, electrical engineering, or a similar degree, with two to three years of practical experience designing computer systems. A person could also have qualified as a POSITA with a combination of (1) more formal education and less technical experience, or (2) less formal education and more technical or professional experience..

22. It is my opinion that such a person would have been actively engaged in learning about the field, possibly through formal instruction and through bibliographic resources. By at least December 2002, such a person would have had access to a vast array of print and electronic resources, including at least the

documents referenced below.

D. Evidence Considered in Forming My Opinions

23. In the preparation of this declaration, I have reviewed **Ex. 0001** and **Ex. 0002** referenced in paragraph 16 above, and any other documents I reference herein. Each of these is a type of material that experts in my field would reasonably rely upon when forming their opinions. I provided a further description of materials I considered in **Appendix C**.

III. Authenticity and Public Availability of *Windows Guidelines Book* (Ex. 0001)

A. Authentication

24. **Ex. 0001** is a true and correct copy of excerpts of *The Windows Interface Guidelines for Software Design*, (“*Windows Guidelines Book*”), published by Microsoft Press in 1995, that I obtained from counsel. This copy is presented as **Ex. 0001** in this declaration.

25. Page 1 of **Ex. 0001** is the front cover that shows the title and publisher and a note that this book “Covers Windows 95 & Windows NT™!” Page 2 is the back cover that shows “1-55615-679-0” as the book’s ISBN (International Standard Book Number). Page 3 is the title page that shows the same title and publisher information as the cover, and page 4 is the copyright page that shows Microsoft Press of Redmond, Washington published this book, which has a “1995” copyright date with Microsoft Corporation as the copyright holder. This page also shows a record of

“Library of Congress Cataloging-in-Publication Data” that lists “95-330” as the Library of Congress Control Number (LCCN) for this record. Pages 5 to 16 (internal pages [iii] to xiv) are the table of contents that shows this book has 14 chapters, five appendices, a glossary, a bibliography and an index. Pages 17 to 20 (internal pages [xvii] to xx) are the introduction. Pages 21 to 60 (internal pages [237] to 276) present Chapter 10 “Integrating with the System” that includes one table and three figures.

26. To authenticate **Ex. 0001**, I searched WorldCat.org, the free web portal, to identify libraries that held this book, and a record at <https://search.worldcat.org/title/32590068> showed the Library of Congress was one of the holding libraries. I then obtained a copy of the front matter of the *Windows Guidelines Book* from the Library of Congress (“LC”). This copy is presented as **Ex. 0001A** in this declaration. Page 1 of **Ex. 0001A** shows the front cover and spine of a bound volume that shows “Windows Interface Guidelines for Software Design” as the title, “QA 76.76 .W56 W553 1995 Copy 2” as the call number, and a label of “FT MEADE GenColl” that indicates this book is in the general collection at the Fort Meade storage facility of LC. Page 2 is the front cover, page 3 is the title page that carries a date stamp of “Library of Congress DEC 15 1995 Copyright Office,” and page 4 is the copyright page that shows a hand-written call number. Pages 5 to 16 (internal pages [iii] to xiv) are the table of contents, page 17 (internal page xx) shows the last page of “Introduction,” and page 18 shows the last numbered page of this

book is page “556.” Page 19 is the back cover. I have closely compared **Ex. 0001** with this copy from LC (**Ex. 0001A**) and found the front and back covers, title page, copyright page, and table of contents in **Ex. 0001** are the same as their counterparts in **Ex. 0001A**.

27. In addition, I have examined the physical copy of the *Windows Guidelines Book* that was used by counsel to produce **Ex. 0001**, and can state with confidence that the pages of **Ex. 0001** are the same as their corresponding pages in the physical copy I reviewed. Furthermore, I have compared **Ex. 0001** with a digital copy of the *Windows Guidelines Book* available from the Internet Archive at <https://archive.org/details/windowsinterface00micr/page/n9/mode/2up>, and found that the pages in **Ex. 0001** are the same as their corresponding pages in this digital copy.

28. The excerpts of the *Windows Guidelines Book* (**Ex. 0001**) are in a condition that creates no suspicion about its authenticity. Specifically, this copy is not missing any intermediate pages (outside of chapters and appendices that are omitted), and there are no visible alterations to the document. Furthermore, the front matter in **Ex. 0001** is the same as the copy I obtained from LC (**Ex. 0001A**). In addition, the pages in **Ex. 0001** are the same as their corresponding pages in a physical copy I have reviewed; and these pages are also the same as their corresponding pages in a digital copy available from the Internet Archive. Based on my review and

analysis, it is my opinion that the pages in **Ex. 0001** are from a book published in 1995 that has an ISBN of “1-55615-679-0” and the Library of Congress holds a copy of this book. For these reasons, it is my opinion that **Ex. 0001** is authentic.

B. Records of the Library of Congress

29. To determine the public availability date of the *Windows Guidelines Book*, I used screen captures to obtain the bibliographic and MARC records for this book from the online catalog of LC. These records are presented as **Ex. 0001B** in this declaration. I personally identified, located, and obtained these records. They are the type of records experts in my field would reasonably rely upon when forming their opinions.

30. Pages 1 to 4 of **Ex. 0001B** are the bibliographic record that shows the title, publisher, copyright date, physical description, notes, ISBNs, and LCCN (“95000330”). These details match the information presented in **Ex. 0001** and **Ex. 0001A**, confirming this record represents the *Windows Guidelines Book*. This record also includes subjects, Library of Congress Classification Number, and Dewey Decimal Classification Number assigned to this book. Page 4 shows LC holds two copies.

31. Pages 5 to 7 are the MARC record. The first six digits of Field 008 show the record was created on “950515” (*i.e.*, May 15, 1995). Field 955 is a field for local cataloging information, and it shows a pre-publication record was created on “05-15-

95” (i.e., May 15, 1995), and the book was received and sent to the CIP (Cataloging-in-Publication Program) for record verification on “09-13-95” (i.e., September 13, 1995). Subfields “a” and “c” of Field 040 show that “DLC” created the original record and entered it into the WorldCat database of OCLC. According to the Directory of OCLC Members (<https://www.oclc.org/en/contacts/libraries.html>), “DLC” is the OCLC library symbol for the Library of Congress. A customary library practice is to process a newly cataloged book promptly for public access, usually within a week after the cataloging work is completed. My personal experience with the Library of Congress collections is that it can take one to three months for a newly cataloged book to be processed for public access. It is therefore my opinion that the *Windows Guidelines Book* became discoverable in the online catalog of LC from May 15, 1995, the record was verified by CIP on September 13, 1995, and the physical copy, after processing, would become publicly accessible no later than December 13, 1995, at this library.

32. Field 010 of the MARC record shows the LCCN, Fields 020 show the 10-digit and 13-digit ISBNs, Field 245 shows the title, and Field 260 shows the publisher and copyright date. Field 300 shows the front matter of this book ends on page xx, the last numbered page of this book is page 556, and the book includes illustrations, some of them in color. The notes in Field 500 match the information on the front cover of this book. Field 504 shows the book includes bibliographical

references and an index. Field 710 shows the publisher is included in the record to help users discover this book.

33. The subjects of the *Windows Guidelines Book* are represented in Field 050 by a Library of Congress Classification (LCC) number of “QA76.76.W56” that represents computer software for “Windows” (source: Library of Congress Classification Schedule Q at https://www.loc.gov/aba/publications/FreeLCC/LCC_Q2020TEXT.pdf), and in Field 082 by a Dewey Decimal Classification (DDC) number of “005.265” that represents “Programming for specific computers” (source: Dewey Decimal Classification. Class 000 Computer Science, Information, General Works at <https://www.oclc.org/content/dam/oclc/webdewey/help/000.pdf>). Subjects are also represented in Field 630 and Fields 650 by three Library of Congress subject headings: “Microsoft Windows (Computer file),” “User interfaces (Computer systems)” and “Computer software \$x Development” (“\$x” indicates topical subheading). These subject representations enable users to conduct subject searches to discover the *Windows Guidelines Book*.

34. This MARC record (**Ex. 0001B**) shows that LC created a pre-publication record for the *Windows Guidelines Book* on May 15, 1995, and verified their cataloging record on September 13, 1995, making this book discoverable in their online catalog from May 15, 1995. By at least September 13, 1995, the interested

public could conduct subject searches by the LCC number, DDC number, and the Library of Congress subject headings in the MARC record to discover this book. They could also search for this book by the LCCN, ISBNs, and title. The physical copy, after processing, would become publicly available by December 13, 1995, at the Library of Congress.

35. Based on my personal experience teaching courses at the Library of Congress in the early 1990s, I know the library had an online catalog on site for users. As evidence that the Library of Congress had an online catalog in operation in 1995 when the *Windows Guidelines Book* became discoverable at this library, I obtained a copy of “LOCIS: Library of Congress Information System” archived by Internet Archive. This record is used to show remote access to the online catalog of LC was available. I personally located, identified, and obtained this profile. This is the type of record experts in my field would reasonably rely upon when forming their opinions. This archived copy is presented as **Ex. 0001C** in this declaration.

36. Internet Archive was founded in May 1996, so web resources that appeared on the Internet before this date are not archived by Internet Archive. **Ex. 0001C** is the earliest available record of remote access to the LC online catalog. Page 1 is a screen capture of a Wayback Machine calendar that shows the LOCIS webpage was archived 75 times between 1997 and 2021, and this particular copy was archived on “MAY 24 1997” and available at

<https://web.archive.org/web/19970524160715/>

<http://library.usask.ca/hytelnet/us3/us373.html>. Page 2 is a PDF of the archived webpage that shows that by May 1997 interested users would be able to telnet into the Library of Congress Information System that includes their catalog. Telnet is a network protocol that supports users to access remote systems on the Internet or other local area networks (source: Telnet at <https://en.wikipedia.org/wiki/Telnet>).

Ex. 0001C shows that by May 1997 interested users were able to log into the Library of Congress online catalog on the Internet to discover the *Windows Guidelines Book*.

37. In addition, I obtained a profile of the Library of Congress at <https://librarytechnology.org/library/2263> from libraries.org, a directory of libraries throughout the world, that provides information on the library's online catalog. I personally located, identified, and obtained this profile. This is the type of record experts in my field would reasonably rely upon when forming their opinions. This profile is presented as **Ex. 0001D** in this declaration.

38. The Technology Profile on page 2 of **Ex. 0001D** shows LC started using Voyager as their online catalog in 1998, and migrated to FOLIO – EBSCO Information Services in 2023. These dates mean that from 1998 on, interested users could use the Voyager system to search for the *Windows Guidelines Book* and access this book at the Library of Congress.

C. Copyright Registration Record

39. **Ex. 0001E** is a true and correct copy of the copyright registration record for the *Windows Guidelines Book*. I personally located, identified, and obtained this record from the Public Records System of the US Copyright Office. This is the type of record experts in my field would reasonably rely upon when forming their opinions.

40. The title, copyright claimant, description, and imprint in **Ex. 0001E** match the information presented in **Ex. 0001** and **Ex. 0001A**, confirming this record represents the *Windows Guidelines Book*. The Date of Publication field shows this book was published on “1995-08-01” (*i.e.*, August 1, 1995) and the “Registration Number / Date” field shows the book was registered with the US Copyright Office on “1995-12-15) (*i.e.*, December 15, 1995). The registration date aligns well with the date stamp of “DEC 15 1995” on the LC copy (**Ex. 0001A**) and the publication date aligns well with LC MARC record completion date of September 13, 1995 (**Ex. 0001B**).

D. Records of the University of Michigan Libraries

41. My research on WorldCat.org also identified the University of Michigan Libraries as another holding library of the *Windows Guidelines Book*. To further demonstrate the availability of this book at libraries, I obtained this library’s bibliographic and MARC records for the *Windows Guidelines Book* from their online

catalog. I personally located, identified, and obtained these records. They are the type of records experts in my field would reasonably rely upon when forming their opinions. These records are presented as **Ex. 0001F** in this declaration.

42. **Ex. 0001F** is a true and correct copy of the bibliographic and MARC records for the *Windows Guidelines Book* that I obtained from the University of Michigan Libraries. Pages 1 and 2 show the bibliographic record for the *Windows Guidelines Book* that includes title, format, contributor, publisher, language, notes, physical description, bibliography, note, ISBN, call number, OCLC number, and subjects. Pages 3 to 5 are the MARC records, and the first six digits of Field 008 show the MARC record was created on “960215” (*i.e.*, February 15, 1996). Subfields “a” and “c” of Field 040 shows DLC (*i.e.*, the Library of Congress) created the original record, and subfield “d” shows “EYM” modified the original record later. The Directory of OCLC Members (<https://www.oclc.org/en/contacts/libraries.html>) shows “EYM” is the OCLC library symbol for the University of Michigan. Fields 008 and 040 inform my opinion that the University of Michigan Libraries used the original record created by the Library of Congress to create their copy cataloging record. The LC MARC record (**Ex. 0001B**) shows the LC original record was created on May 15, 1995, and verified and completed on September 13, 1995. The date in Field 008 of the MARC record of the University of Michigan Libraries shows the completion date for their copy cataloging record is February 15, 1996. A customary

library practice is to process a newly cataloged book promptly for public access, usually within a week after the cataloging work is completed. It is therefore my opinion that the *Windows Guidelines Book* became discoverable at the University of Michigan Libraries from February 15, 1996 on, and the physical copy, after processing, became publicly available at this library no later than February 22, 1996.

43. Since **Ex. 0001F** is a copy cataloging record based on the LC MARC record (**Ex. 0001B**), these two records have many data elements in common, including the LCCN in Field 010, the 10-digit ISBN in Field 020, title in Field 245, publisher information in Field 260, physical description in Field 300, notes in Field 500 and Field 504, and the publisher in Field 710. These two records also have the same subject representations, including the Library of Congress Classification number in Field 050, and Library of Congress subject headings in Field 630 and Fields 650. Except for data elements specific to each library, these two records are nearly identical.

44. The MARC record of the University of Michigan Libraries was completed on February 15, 1996, making the *Windows Guidelines Book* discoverable in their online catalog from that date on. Interested users would be able to conduct subject searches to discover this book. They would also be able to search for this book by LCCN, ISBN, title, and publisher. The physical copy, after processing, would become publicly available by February 22, 1996 at this library.

45. As evidence that the University of Michigan Libraries had an online catalog in operation in 1996 when the *Windows Guidelines Book* became discoverable at this library, I obtained a profile of the library at <https://librarytechnology.org/library/2089> from libraries.org, a directory of libraries throughout the world. I personally located, identified, and obtained this profile. This is the type of record experts in my field would reasonably rely upon when forming their opinions. This profile is presented as **Ex. 0001G** in this declaration.

46. The Technology Profile on page 2 of **Ex. 0001G** shows the University of Michigan Libraries started using NOTIS as their online catalog in 1987, and migrated to ALEPH 500 in 2002. These dates mean that in 1996 interested users would be able to use NOTIS to search for the *Windows Guidelines Book* and access this book at this library.

E. Internet Archive Record

47. To determine when the *Windows Guidelines Book* became available for online purchase, I searched the Internet Archive for records. **Ex. 0001H** is a true and correct copy of the publisher's webpage for this book. I personally located, identified, and obtained this record. It is the type of record experts in my field would reasonably rely upon when forming their opinions.

48. Page 1 of **Ex. 0001H** is a screen capture of a Wayback Machine calendar that shows this webpage was archived four times between "22 Apr 2000"

and “16 Jun 2000” and this particular copy was archived on “APR 22 2000” and available at <https://web.archive.org/web/20000422051029/http://www1.fatbrain.com/asp/bookinfo/bookinfo.asp?theisbn=1556156790>. The archive date of April 22, 2000 means this webpage was publicly discoverable on the Internet no later than this date.

49. Page 2 is the archived webpage saved in PDF that shows the title, list price, pagination, publisher, date of publication (“07/1995”), ISBN (“1556156790”), and a summary of the *Windows Guidelines Book*. It also shows the online price, which is 20% off the list price, and a link for users to “Add Item to Basket.” **Ex. 0001H** informs my opinion that the *Windows Guidelines Book* became available for online purchase no later than April 22, 2000.

F. Usage Records

50. Actual usage of a publication is reflected by other publications that make reference to it. I have reviewed the citation history of the *Windows Guidelines Book* on the ACM Digital Library to compile a list of early citations. **Ex. 0001I** presents seven citations published from 1996 and 1997, and shows the earliest citations appeared in April 1996.

G. Summary of My Opinion on the *Windows Guidelines Book*

51. It is my opinion that **Ex. 0001** is authentic because it contains no signs to raise suspicion about its authenticity, and the content of this copy is the same as

the physical copy I received from counsel and the digital copy I reviewed on the Internet Archive. The front matter of **Ex. 0001** is also the same as the front matter in the copy from Library of Congress (**Ex. 0001A**). The copyright registration record (**Ex. 0001E**) shows the *Windows Guidelines Book* was published on August 1, 1995. Bibliographic and MARC records from the Library of Congress (**Ex. 0001B**) show the library created a pre-publication record for this book on May 15, 1995, and completed the cataloging record on September 13, 1995. The physical copy, after processing, would become publicly available no later than December 13, 1995. Bibliographic and MARC records from the University of Michigan Libraries (**Ex. 0001F**) show the library used the LC record for copy cataloging and completed their copy cataloging record on February 15, 1996, making this book discoverable from that date on at this library, and the physical copy, after processing, would become publicly available by February 22, 1996. Citation history (**Ex. 0001I**) shows the earliest citations appeared in April 1996, supporting my opinion that the *Windows Guidelines Book* became publicly accessible at the Library of Congress by December 13, 1995, and became publicly accessible at the University of Michigan Libraries by February 22, 1996. The publisher's webpage for this book shows it became available for online purchase by April 22, 2000.

52. Based on the documents and records I have reviewed, it is my opinion that *The Windows Interface Guidelines for Software Design* became publicly

discoverable and accessible at the Library of Congress by December 13, 1995, and became publicly accessible at the University of Michigan Libraries by February 22, 1996. The earliest citations published in April 1996 support my opinions. The publisher's webpage for this book shows it was available for online purchase no later than April 2000.

IV. Authenticity and Public Availability of *Gomberg* (Ex. 0002)

A. Authentication

53. **Ex. 0002** is a true and correct copy of “A comparison of large-scale software installation methods on NT and UNIX,” (“*Gomberg*”), by Gomberg, et al. in *Proceedings of the Large Installation System Administration of Windows NT Conference, August 5-8, 1998, Seattle, Washington, USA*, that I obtained from the USENIX legacy website at https://www.usenix.org/legacy/publications/library/proceedings/lisa-nt98/full_papers/gomberg/gomberg.pdf. This copy is presented as **Ex. 0002** in this declaration.

54. Page 1 is a cover sheet that shows the title and authors of *Gomberg*, and indicates this paper “was originally published in the Proceedings of the Large Installation System Administration of Windows NT Conference Seattle, Washington, August 5-8, 1998.” Pages 2 to 12 are the full text of *Gomberg*, and Page 2 shows the title, authors, and abstract. The pages in this copy are not numbered. **Ex. 0002** shows this paper has 10 sections and an appendix of tools, including NT Resource Kit, NT

Rollout tools, NT Services, Other Microsoft Tools, Windows Development Tools, and Third party. This paper includes one figure and three references.

55. To authenticate **Ex. 0002**, I searched WorldCat.org, the free web portal, for libraries holding the conference proceedings containing *Gomberg*, and a record at <https://search.worldcat.org/title/39953370> showed 19 libraries held the print version of *Proceedings of the Large Installation System Administration of Windows NT Conference: August 5-8, 1998, Seattle, Washington* (“LISA-NT 98”). I reviewed the library records for the proceedings and identified the Pennsylvania State University Libraries (“Penn State Libraries”) as the creator of the original record. I then requested a copy of *Gomberg* from the Penn State Libraries via a document vendor. This library copy is presented as **Ex. 0002A** in this declaration.

56. Page 1 of **Ex. 0002A** is the cover that shows this publication is the conference proceedings of LISA-NT 1998, and the USENIX Association and the System Administrators Guild are the conference sponsors. It also shows a call number label of Math TK5105.5 .L375 1998. Page 2 is the title page. Page 3 is the table of contents that shows *Gomberg* is included in the session on “Operating System and Software Installation” scheduled for Friday, August 7, and the paper begins on page 37. Pages 4 to 14 (internal pages 37 to 47) are the full text of *Gomberg*, and all the pages are identified by a footer “USENIX Association. Large Installation System Administration of Windows NT Conference.” Page 15 is the back cover that shows a

barcode of the Penn State Libraries and an ISBN of “1-880446-96-0.”

57. I have closely compared **Ex. 0002** with the Penn State copy (**Ex. 0002A**), and found the two copies to have the same content. The differences are that the pages of the digital version (**Ex. 0002**) are not numbered, and the numbered pages of the print version (**Ex. 0002A**) have a footer identifying the conference where *Gomberg* was presented, and the conference organizer (“USENIX Association”).

58. **Ex. 0002** is in a condition that creates no suspicion about its authenticity. Specifically, the text appears to flow seamlessly, and there are no visible alterations to the pages. Based on my analysis, it is my opinion that **Ex. 0002** is an authentic copy of *Gomberg*.

B. Records of the Penn State University Libraries

59. **Ex. 0002B** is a true and correct copy of the bibliographic and MARC records for the *Proceedings of the Large Installation System Administration of Windows NT Conference: August 5-8, 1998, Seattle, Washington* (“LISA-NT 1998”). I personally located, identified, and obtained these records from the online catalog of the Penn State University Libraries. They are the type of records experts in my field would reasonably rely upon when forming their opinions.

60. Pages 1 and 2 are the bibliographic record that shows the title, conference author, publisher, copyright date, physical description, and additional creators. It also shows the book’s call number and its location. In addition, the record

shows subjects, ISBN, note on sponsors, and a bibliography note.

61. Pages 3 and 4 are the MARC record. The first six digits of Field 008 show the record was created on “980910” (i.e., September 10, 1998). Subfields “a” and “c” of Field 040 show “PSt” created and entered the record into the WorldCat database of OCLC. According to the MARC Code List for Organizations (<https://www.loc.gov/marc/organizations/org-search.php>), “PSt” is the MARC code for the Pennsylvania State University. Fields 008 and 040 inform my opinion that the Pennsylvania State University library system created an original MARC record for LISA-NT 1998 on September 10, 1998, making this work discoverable in the library system’s online catalog from that date on. A customary library practice is to process a newly cataloged monograph promptly for public access, usually within one week after the cataloging work is completed. It is therefore my opinion that LISA-NT 1998 that contains *Gomberg* became publicly discoverable at the Penn State University Libraries on September 10, 1998, and the physical copy, after processing, became available for public access by September 17, 1998.

62. Field 020 shows the ISBN of this work. Field 111 shows the conference name is used as the author. Field 245 shows the title and statement of responsibility. Fields 264 shows USENIX Association of Berkeley, California, published this work with a copyright date of 1998. Field 300 shows this work has 88 pages and includes illustrations. Field 500 shows the conference has two sponsors, and they are included

in Fields 710 to help users discover this work.

63. The subjects of LISA-NT 1998 are represented in Field 050 by a Library of Congress Classification (LCC) number of “TK5105.5” that represents general works on “Computer networks” (source: Library of Congress Classification Schedule T at https://www.loc.gov/aba/publications/FreeLCC/LCC_T2019TEXT.pdf), and in Fields 630 and Fields 650 by four Library of Congress subject headings, with “Microsoft Windows NT,” “UNIX (Computer file),” “Computer networks x| Management” (“x|” indicates a topical subheading), and “Large scale systems” as the main topic, each followed by “Congresses” as a form subheading to indicate the main topics are treated at a conference. These subject representations enable users to conduct subject searches to discover LISA-NT 1998 that contains *Gomberg*.

64. As evidence that the Penn State University Libraries had an online catalog in operation in 1998 when LISA-NT 1998 became accessible at the library system, I used screen capture to obtain an archived record of their access system. I personally located, identified, and obtained this record from Internet Archive. This is the type of record experts in my field would reasonably rely upon when forming their opinions. This record is presented as **Ex. 0002C** in this declaration.

65. The top of **Ex. 0002C** is a Wayback Machine calendar that shows this website was archived 284 times between “18 Oct 1996” and “29 Mar 2008” and this particular copy was archived on “FEB 09 1998” (*i.e.*, February 9, 1998) and available

at <https://web.archive.org/web/19980209162244/http://www.lias.psu.edu/>. The title of this website is “LIAS. Penn State Libraries Information Access System” and it shows that as of February 9, 1998, interested users had several ways to explore various types of resources at the Penn State library system, and that for books and more, interested users should use “The CAT.” **Ex. 0002C** confirms that by February 9, 1998, the Penn State library system had an information access system in operation, including an online catalog for users to search for books and more.

C. Records of the University of Minnesota Libraries

66. My research on WorldCat.org, the free web portal, also identified the University of Minnesota Libraries as another holding libraries of the conference proceedings that contain *Gomberg*. To further demonstrate the availability of LISA-NT 1998 at libraries, I obtained bibliographic and MARC records for this work from the online catalog of the University of Minnesota Libraries. I personally located, identified, and obtained these records. They are the type of records experts in my field would reasonably rely upon when forming their opinions. These records are presented as **Ex. 0002D** in this declaration.

67. Pages 1 to 4 are the bibliographic record that shows the title, responsible organization, conference name, sponsors, publisher, copyright date, physical description, language, notes, subjects, genre, ISBN and OCLC number.

68. Page 5 is the MARC record, and the first six digits of Field 008 show

the record was created on “980910” (*i.e.*, September 10, 1998). Subfields “a” and “c” of Field 040 show “UPM” created the original record and entered it into the WorldCat database on OCLC, and subfield “d” shows “MnU” modified the record later. According to the Directory of OCLC Members (<https://www.oclc.org/en/contacts/libraries.html>), “UPM” is the OCLC library symbol for the Pennsylvania State University, and “MnU” is the OCLC library symbol for the University of Minnesota, Minneapolis. Field 945 is a field for local cataloging information and it shows the record was completed on “1998-10-14” (*i.e.*, October 14, 1998). Fields 008, 040, and 945 inform my opinion that the Penn State University Libraries created the original record for LISA-NT 1998 that contains *Gomberg* on September 10, 1998, and the University of Minnesota Libraries used that original record for copy cataloging and completed their record on October 14, 1998, making LISA-NT 1998 discoverable in their online catalog from that date on. The physical copy, after processing, would become publicly accessible at this library system no later than October 21, 1998.

69. Since **Ex. 0002D** is a copy cataloging record based on the Penn State MARC record (**Ex. 0002B**), these two records have many data elements in common, including the 10-digit ISBN in Field 020, title in Field 245, physical description in Field 300, notes in Field 500 and Field 504, and the conference sponsors in Fields 710. These two records also have the same subject representations, including

the Library of Congress Classification number in Field 050, and Library of Congress subject headings in Fields 630 and Fields 650. One difference is that **Ex. 0002B** shows the University of Michigan Libraries followed the latest cataloging encoding practice to record publisher and copyright date information in Fields 264, while **Ex. 0002D** shows that the University of Minnesota Libraries records the publisher and copyright date information in Field 260. Except for data elements specific to each library, these two records are nearly identical.

70. The MARC record of the University of Minnesota Libraries was completed on October 14, 1998, making LISA-NT 1998 that contains *Gomberg* discoverable in their online catalog from that date on. Interested users would be able to conduct subject searches to discover this book. They would also be able to search for this book by ISBN, conference name, title, and conference sponsors. The physical copy, after processing, would become publicly available by October 21, 1998, at this library.

71. As evidence that the University of Minnesota Libraries had an online catalog in operation in 1998 when LISA-NT 1998 became discoverable at this library, I obtained a profile of the library at <https://librarytechnology.org/library/1024> from libraries.org, a directory of libraries throughout the world. I personally located, identified, and obtained this profile. This is the type of record experts in my field would reasonably rely upon when forming their opinions. This profile is presented as

Ex. 0002E in this declaration.

72. The Technology Profile on page 2 of **Ex. 0002E** shows the University of Minnesota Libraries started using NOTIS as their online catalog in 1986, and migrated to ALEPH 500 in 2002. These dates mean that in 1998 interested users would be able to use NOTIS to search for LISA-NT 1998 and access this book and (*Gomberg* therein) at this library.

D. Internet Archive Records

73. Internet Archive is a digital library that archives web resources, scholarly works, cultural artifacts, and other resources to make the materials freely accessible for the general public. Internet Archive usually does not archive a web file as soon as the file appears on the Internet, meaning there is usually a time lag between the appearance of a web file and the archiving of that file by Internet Archive. The archive date of a web file archived by Internet Archive therefore means that the archived web file is publicly accessible before, and no later than, the archive date.

74. To determine the public availability of *Gomberg* on the Internet, I searched Internet Archive for records. I have obtained an archived copy of the USENIX Conference Proceedings website, an archived copy of the Technical Program of LISA-NT 1998, and an archived copy of the webpage for *Gomberg* abstract from the Internet Archive. I personally located, identified, and obtained these records. They are the type of records experts in my field would reasonably rely upon

when forming their opinions. These archived copies are presented as **Ex. 0002F**, **Ex. 0002G**, and **Ex. 0002H**, respectively, in this declaration.

75. **Ex. 0002F** is a true and correct copy of the website of USENIX Conference Proceedings (by Year and Location). Page 1 is a screen capture of a Wayback Machine calendar that shows this website was archived 134 times between 1996 and 2025, and this particular copy was archived on “JAN 16 1999” and available at <https://web.archive.org/web/19990116232824/http://www.usenix.org/publications/library/proceedings/index.html>. Pages 2 to 4 are the PDF of the archived website that shows “Large Installation System Administration of Windows NT Conference” is a 1998 conference. The top of this website explains that “[a]bstracts to papers are available to everyone. The full papers are available only to USENIX members. When accessing papers, remember to use your *member number* of your *userid*. To become a USENIX member, please see our Membership Information.” (emphasis in original). **Ex. 0002F** shows that by January 16, 1999, interested users would be able to discover the USENIX Conference Proceedings website on the Internet and learn that abstracts of the 1998 Large Installation System Administration of Windows NT Conference were available to everyone, the full papers were available only to USENIX members, and non-members could become members to have access to full papers.

76. **Ex. 0002G** is a true and correct copy of the Technical Program website

of LISA-NT 1998. Page 1 is a screen capture of a Wayback Machine calendar that shows this website was archived 36 times between 1999 and 2011, and this particular copy was archived on “AUG 23 1999” and available at <https://web.archive.org/web/19990823080333/http://www.usenix.org/publications/library/proceedings/lisa-nt98/technical.html>. Pages 2 and 3 are the PDF of the archived website that shows *Gomberg* was presented on Friday, August 7, at the session on “Operating System and Software Installation.” **Ex. 0002G** shows that by August 23, 1999, interested users would discover the website of the Technical Program of LISA-NT 1998 and use it to access *Gomberg*.

77. **Ex. 0002H** is a true and correct webpage for *Gomberg*’s abstract. Page 1 is a screen capture of a Wayback Machine calendar that shows this webpage was archived 48 times between 1999 and 2024, and this particular copy was archived on “NOV 13 1999” and available at <https://web.archive.org/web/19991113081156/http://www.usenix.org/publications/library/proceedings/lisa-nt98/gomberg.html>. Page 2 is the PDF of the archived webpage that shows the title, authors, and abstract of *Gomberg* and indicates that “USENIX Members may view the full text of this paper in HTML form and PDF form” and need to use their membership number as the username for access. It also indicates that users interested in becoming USENIX members should visit “Membership Information.” **Ex. 0002H** shows that interested users could discover this webpage about *Gomberg*’s abstract by November 13, 1999

on the Internet, and USENIX members would be able to access the full text no later than that date.

78. In short, **Ex. 0002F** shows that by January 16, 1999, interested users would be able to discover on the Internet that the USENIX Conference Proceedings website included the LISA-NT 1998 conference and that the conference abstracts were available to all interested users, while the full papers were available only to USENIX members. **Ex. 0002G** shows that interested users would be able to discover the Technical Program website of LISA-NT 1998 by August 23, 1999, and learn from the website that *Gomberg* was included in this conference. **Ex. 0002H** shows that by November 1999, interested users would discover the webpage for *Gomberg*'s abstract and use it to access the full paper if they were USENIX members.

E. Usage Records

79. Actual usage of a publication is reflected by other publications that make reference to it. I used Google Scholar and Semantic Scholar to review citations to *Gomberg* and compile a list of citations published in 2000 and 2001. **Ex. 0002I** presents six citations to *Gomberg*, showing the earliest citation was published in April 2000.

F. Summary of My Opinion on *Gomberg*

80. It is my opinion that the copy of *Gomberg* (**Ex. 0002**) obtained from the USENIX legacy server is authentic because the article contains no signs to raise

suspicion about its authenticity, and this copy is identical to the copy obtained from a print version held by the Pennsylvania State University Libraries (**Ex. 0002A**).

81. The bibliographic and MARC records of the Penn State University Libraries for LISA-NT 1998 that contains *Gomberg* (**Ex. 0002B**) show the library system created an original record for the conference proceedings on September 10, 1998, making this work discoverable in their online catalog. The physical copy, after processing, would become publicly accessible by September 17, 1998, at this library system. The bibliographic and MARC records of the University of Minnesota Libraries for this work (**Ex. 0002D**) show the library system used the Penn State MARC record for copy cataloging and completed their copy cataloging record on October 14, 1998, making this work discoverable in their online catalog from that date on. The physical copy, after processing, would become publicly accessible by October 21, 1998, at this library system. Archived copies of the USENIX Conference Proceedings website, the website of the Technical Program of LISA-NT 1998, and the webpage for *Gomberg*'s abstract (**Exs. 0002F-H**) show that interested users would be able to discover information about LISA-NT 1998 on the Internet by January 1999 and follow the links to access *Gomberg*. USENIX members would be able to access the full paper, and non-members could become members to access the full paper. Citation history shows the earliest citation was published in April 2000.

82. Based on the documents and records I have reviewed, it is my opinion

that the conference proceedings containing *Gomberg* were publicly accessible at the Pennsylvania State University Libraries no later than September 17, 1998, and publicly accessible at the University of Minnesota Libraries no later than October 21, 1998. It is also my opinion that *Gomberg* was available to USENIX members by January 1999 because interested users would discover the conference and *Gomberg* from the USENIX Conference Proceedings website archived on January 16, 1999.

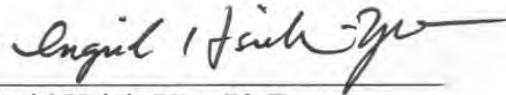
V. Conclusion

83. In signing this Declaration, I recognize that the Declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in this proceeding. If required, I will appear for cross-examination at the appropriate time.

I hereby declare that all statements made herein of my own knowledge are true and that all statements are made on information and belief are believed to be true, and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. §1001.

Dated: 10-7-2025

Respectfully submitted,

A handwritten signature in black ink, appearing to read 'Ingrid Hsieh-Yee', written over a horizontal line.

Ingrid Hsieh-Yee, Ph.D.
Annapolis, Maryland

APPENDIX A

Ingrid Hsieh-Yee
Professor Emerita
Dept. of Library and Information Science
Catholic University of America
Washington, D.C. 20064
E-mail: hsiehyee@cua.edu

Education

Ph.D. 1990. Library and Information Studies, University of Wisconsin-Madison
Minors: Sociology and Psychology

M.A. 1984. Library and Information Studies, University of Wisconsin-Madison.

M.A. 1983. Comparative Literature, University of Wisconsin-Madison.

B.A. 1980. Foreign Languages and Literature, National Taiwan University.

Work Experience

Professor, School/Dept. of Library and Information Science, Catholic University of America,
1990-2022 (Assistant Professor, 1990-1996; Associate Professor, 1997-2004; Professor
2004-2022; Professor Emerita 2022)

Co-Chair, Dept. of Library and Information Science, Catholic University of America, June 2015-
August 2016.

Acting Dean, School of Library and Information Science, Catholic University of America,
January 2010-June 2012.

Cataloger, Dept. of Legislative Reference Library, Annapolis, Maryland, 1989-1990.

Lecturer, School of Library and Information Studies, University of Wisconsin-Madison, 1988.

Teaching Assistant, School of Library and Information Studies, University of Wisconsin-
Madison, 1986-1988.

Cataloger, Health Sciences Library, University of Wisconsin-Madison, 1984-1986.

Areas of Teaching and Research Interests

Information Organization and Access; Metadata; Cataloging & Classification; Information
Architecture; Information Retrieval; Digital Collections; Social Media; Scholarly

Communication; Information Behavior; Health Informatics; Human Computer Interaction; Usability Studies

Grants & Honors

Cultural Heritage Information Management Project. IMLS grant. Amount: \$498,741. Period: Aug. 2012 to July 2015. Co-PI with Dr. Youngok Choi.

D.C. Health Information Technology (HIT4): Building Capacity & Providing Access in Our Nation's Capital. Dept. of Labor H2B Training Grant. Grant amount: \$4,175,500. Grant period: Nov. 2011 to Dec. 2015. Partner with the Metropolitan School of Professional Studies of the Catholic University of America, Children's National Medical Center, D.C. Department of Employment Services, Holy Cross Hospital, Howard University, Center for Urban Progress, Providence Hospital, and Sibley Memorial Hospital.

Capital Health Careers Project. Department of Labor Healthcare Sector and Other High Growth and Emerging Industries Grant. Grant amount: \$4,953,999. Grant period: March 2010 – February 2013. Awarded to a group of healthcare organizations and educational institutions in Washington, D.C. Providence Health Foundation of Providence Hospital (Lead institution). Part of the grant supported the development of a Master's degree program in Information Technology with a concentration in Health Information Technology offered by the School of Library and Information Science.

The Washington D.C. School Librarians Project. IMLS grant. Grant amount: \$412,660. Grant period: Aug. 2007 – June 2011. The School partnered with the District of Columbia Public Schools (DCPS) and the District of Columbia Library Association to educate and mentor school media specialists for the DCPS system. PI, Jan. 2010 to June 2011.

SIG Member of the Year, American Society for Information Science and Technology (2009).

Most Outstanding Paper of *OCLC Systems & Services* (2001).

ALISE Research Grant (2001).

Most Outstanding Paper of *OCLC Systems & Services* (2000).

Research Grant from ERIC (1999-2000).

Best Research Paper Award; Association for Library and Information Science Education (1998).

Research Grants, Catholic University of America. 1991, 1992, 1993, 1996, 1998, 1999, 2004, 2005, 2006, 2007, 2013-14.

Cooperative Faculty Research Grant, Consortium of Universities in the Washington Metropolitan Area (1993-1994).

Cooperative Research Grant, Council on Library Resources (1993-1994).

Journal of the American Society for Information Science Best Paper Award (1993).

ASIS/ISI Information Science Doctoral Dissertation Scholarship (1989).

HEA Title IIB Fellowship (Dept. of Education) (1989)

Chinese-American Librarians Association Scholarship (1987).

Beta Phi Mu (1985).

Vilas Fellowship, University of Wisconsin-Madison. 1984

Publications

Hsieh-Yee, I. (2021). Can We Trust Social Media? *Internet Reference Services Quarterly*, 1-15.
Published online: 28 July, 2021. 10.1080/10875301.2021.1947433.

Bailey, T., & Hsieh-Yee, I. (2020). Combating the Sharing of False Information: History, Framework & Literacy Strategies, *Internet Reference Services Quarterly*, 24 (1-2), 9-30.

Choi, Y., and Hsieh-Yee, I. (2010). Finding Images in an OPAC: Analysis of User Queries, Subject Headings, and Description Notes. *Canadian Journal of Information and Library Science*, 34(3): 271 – 295.

Hsieh-Yee, I. (2008). Educating Cataloging Professionals in a Changing Information Environment. *Journal of Education for Library and Information Science*, 46(2): 93-106.

Vellucci, S. L., Hsieh-Yee, I., and Moen, W.E. (2007). The Metadata Education and Research Information Commons (MERIC): A Collaborative Teaching and Research Initiative. *Education for Information*, 25(3&4): 169-178.

NISO Framework for Guidelines for Building Good Digital Collections. 3rd ed. Baltimore, MD: National Information Standards Organization, 2007. Also available online: <http://www.niso.org/framework/framework3.pdf> (NISO Working Group members: Priscilla Caplan (chair), Grace Agnew, Murtha Baca, Tony Gill, Carl Fleischhauer, Ingrid Hsieh-Yee, Jill Koelling, and Christie Stephenson.)

Choi, Y., Hsieh-Yee, I., and Kules, B. (2007). Retrieval Effectiveness of TOC and LCSH. *Proceedings of the Joint Conference on Digital Libraries*, pp. 233-234.

Vellucci, S., and Hsieh-Yee, I. (2007). They Didn't Teach Me That in Library School! Building

a Digital Teaching Commons to Enhance Metadata Teaching, Learning and Research. *Proceedings of the National Conference of the Association of College and Research Libraries, Baltimore, MD*, pp. 26-31.

Mitchell, Vanessa, and Ingrid Hsieh-Yee. (2007). Converting Ulrich's Subject Headings to FAST Headings: A Feasibility Study. *Cataloging & Classification Quarterly*, 45(1): 59-85.

Hsieh-Yee, I., Tang, R., and Zhang, S. (2007). User Perceptions of a Federated Search System. *IEEE Technical Committee on Digital Libraries Bulletin*, Summer 3(2) (URL = <http://www.ieee-tcdl.org>).

Tang, R., Hsieh-Yee, I., and Zhang, S. (2007). User Perceptions of MetaLib Combined Search: An Investigation of How Users Make Sense of Federated Searching." *Internet Reference Services Quarterly*, 12(12): 211-236.

Hsieh-Yee, I., Tang, R., and Zhang, S. (2006). User Perceptions of a Federated Search System. *Proceedings of the Joint Conference on Digital Libraries, June 11-15, 2006, Chapel Hill*, p. 338.

Hsieh-Yee, I. (2006). *Organizing Audiovisual and Electronic Resources for Access: A Cataloging Guide*. 2nd ed. Westport, Conn.: Libraries Unlimited.

NISO A Framework of Guidance for Building Good Digital Collections. 2nd ed. Bethesda, MD: National Information Standards Organization, 2004. Framework Advisory Group: Grace Agnew, Liz Bishoff, Priscilla Caplan (Chair), Rebecca Gunther and Ingrid Hsieh-Yee.

Hsieh-Yee, I. (2004). Cataloging and Metadata Education in North American LIS Programs. *Library Resources & Technical Services*, 48(1): 59-68.

Hsieh-Yee, I. (2004). Cataloging and Metadata Education. In Gary E. Gorman (Ed.), *International Yearbook of Library and Information Management 2003: Metadata Applications and Management*, (pp.204-234). London: Facet Publishing.

Yee, P. L., Hsieh-Yee, I., Pierce, G.R., Grome, R., and Schantz, L. (2004). Self-Evaluative Intrusive Thoughts Impede Successful Searching on the Internet. *Computers in Human Behavior*, 20(1): 85-101.

Hsieh-Yee, I. (2003). Cataloging and Metadata Education: A Proposal for Preparing Cataloging Professionals of the 21st Century. A report submitted to the ALCTS-Education Task Force in response to Action Item 5.1 of the *Bibliographic control of Web Resources: A Library of Congress Action Plan*. Approved by the Association for Library Collections and Technical Services. Web version available since April 2003 at <http://lcweb.loc.gov/catdir/bibcontrol/CatalogingandMetadataEducation.pdf>.

- Hsieh-Yee, I. (2002). Cataloging and Metadata Education: Asserting a Central Role in Information Organization. *Cataloging & Classification Quarterly* 34(½): 203-222.
- Hsieh-Yee, I., and Smith, M. (2001). The CORC Experience: Survey of Founding Libraries, Part I. *OCLC Systems & Services*, 17: 133-140. (Received "The Most Outstanding Paper of OCLC Systems & Services in 2001" award.)
- Hsieh-Yee, I., and Smith, M. (2001). The CORC Experience: Survey of Founding Libraries, Part II, Automated Tools and Usage. *OCLC Systems & Services*, 17: 166-177. (Received "The Most Outstanding Paper of OCLC Systems & Services in 2001" award.)
- Hsieh-Yee, I. (2001). ERIC User Services: Changes and Evaluation for the Future. *Government Information Quarterly*, 18: 31-42.
- Hsieh-Yee, Ingrid. (2001). Research on Web Search Behavior. *Library and Information Science Research*, 23: 167-185.
- Logan, E., and Hsieh-Yee, I. (2001). Library and Information Science Education in the Nineties. *Annual Review of Information Science and Technology*, 35: 425-477.
- Hsieh-Yee, I. (Ed.) (2001). *Library and Information Science Research*, 23 (2). A special issue in honor of the retirement of Douglas L. Zweizig.
- Hsieh-Yee, I. (2000). *ERIC User Services: Evaluation in a Decentralized Environment*. Washington, D.C.: Dept. of Education.
- Hsieh-Yee, Ingrid. (2000). *Organizing Audiovisual and Electronic Resources for Access: A Cataloging Guide*. Littleton, CO: Libraries Unlimited.
- Hsieh-Yee, I. (2000). Organizing Internet Resources: Teaching Cataloging Standards and Beyond. *OCLC Systems & Services*, 16: 130-143. (Received "The Most Outstanding Paper of OCLC Systems & Services in 2000" award.)
- Hsieh-Yee, I. (1998). The Retrieval Power of Selected Search Engines: How Well Do They Address General Reference Questions and Subject Questions? *Reference Librarian*, 60: 27-47.
- Hsieh-Yee, I. (1998). Search Tactics of Web Users in Searching for Texts, Graphics, Known Items and Subjects: A Search Simulation Study. *Reference Librarian*, 60: 61-85. (Received the 1997 Best ALISE Research Paper Award.)
- Hsieh-Yee, I. (1997). Access to OCLC and Internet Resources: LIS Educators' Views and Teaching Practices. *RQ*, 36: 569-86.

- Hsieh-Yee, I. (1997). Teaching Online and CD-ROM Resources: LIS Educators' Views and Practices. *Journal of Education for Library and Information Science*, 38: 14-34.
- Hsieh-Yee, I. (1996). The Cataloging Practices of Special Libraries and Their Relationship with OCLC. *Special Libraries*, 87: 10-20.
- Hsieh-Yee, I. (1996). Modifying Cataloging Practice and OCLC Infrastructure for Effective Organization of Internet Resources. In *Proceedings of the OCLC Internet Cataloging Colloquium*. [Online]. Available: <http://www.oclc.org/oclc/man/colloq/hsieh.htm>
- Hsieh-Yee, I. (1996). Student Use of Online Catalogs and Other Information Channels. *College & Research Libraries*, 57: 161-175.
- Hsieh-Yee, I. (1995). Ten entries in James S. C. Hu (Ed.), *Encyclopedia of Library & Information Science*, 913, 1028-29, 1036, 1037, 1145-46, 1514, 1575, 1763-64, 2216-27, 2378-79. Taipei, Taiwan: Sino-American Publishing. (Topics include "Advanced Technology/Libraries," "Information Ethics," "Instruction on Cataloging and Classification," "Instruction on Reference Services.")
- Hsieh-Yee, I. (1993). Effects of Search Experience and Subject Knowledge on Online Search Behavior: Measuring the Search Tactics of Novice and Experienced Searchers. *Journal of the American Society for Information Science*, 44: 161-174. (Received the 1993 Best JASIS Paper Award.)

Presentations

- Bailey, T. & Hsieh-Yee, I. (February 2020). The Phenomena of Sharing Misinformation and the Need for Information Literacy. Presented at the 2020 Bridging the Spectrum Symposium, Washington, D.C., February 14, 2020.
- Hsieh-Yee, I. and Fragan-Fly, J. (May 2018) Trends, Design & Strategies for Digital Scholarship Services. Presented at the 2018 Maryland/Delaware Library Association Conference, Cambridge, MD.
- Hsieh-Yee, I. (February, 2018) Research Data Management: What It Takes to Succeed. Presented at the 10th Bridging the Spectrum Symposium, Washington, D.C.
- Hsieh-Yee, I. (February, 2017) *Research Data Management: New Competencies and Opportunities for Information Professionals*. Presented at the 9th Bridging the Spectrum Symposium, Washington, D.C.
- Hsieh-Yee, I. and Lawton, P. (February, 2017) *Enhancing Catholic Portal Searches with User Terms and LCSH*. Presented at the 9th Bridging the Spectrum Symposium, Washington, D.C.

- Hsieh-Yee, I. (2016, October) *Visualizing Data for Information*. Presented at the 2016 Virginia Library Association Conference, Hot Springs, VA.
- Hsieh-Yee, I. (2016, August) *Religious Materials Toolbox for Archivists: Solutions to Problems Facing the Profession*. Presented at Archives * Records 2016, Atlanta, GA.
- Hsieh-Yee, I. and Lawton, P. (2016, March) *Enhancing Retrieval of Catholic Materials with LCSH Knowledge Structure*. Presented at the 2016 Catholic Library Association Conference, San Diego, CA.
- Fagan-Fry, J. and Hsieh-Yee, I. (2016, February) *Approaches to Digital Scholarship at Top Universities around the World: Scholarly Publishing in the Digital Age*. Presented at the 8th Bridging the Spectrum Symposium, Washington, D.C.
- Hsieh-Yee and Fagan-Fry, J. (2016, January) *Innovative Services for Digital Scholarship at Top 100 Research Libraries of the World*. Poster presented at the 2016 Annual Conference of the Association for Library and Information Science Education, Boston, Mass.
- Hsieh-Yee, I. and Lawton, P. (2015, June). *Crowdsourcing terms for CRRRA portal themes*. Poster presented at the third CRRRA symposium and annual meeting, Bringing the created toward the Creator: Liturgical art and design since Vatican II. Catholic Theological Union, Chicago, Illinois.
- Hsieh-Yee, I. and Lawton, P. (2015, February). *Crowdsourcing terms for thematic exploration in the Catholic Portal*. Poster presented at the 7th Annual Bridging the Spectrum Symposium, Washington, D.C.
- Hsieh-Yee, I., James, R., and Fagan-Fry, J. (2015, February). *Support for digital scholarship at top university libraries of the world*. Poster presented at the 7th Annual Bridging the Spectrum Symposium, Washington, D.C.
- Hsieh-Yee, I., Zhang, S., Lin, K., and Cherry, S. (2015, February). *Thus said the end users: Summon experience and support for research workflows*. Poster presented at the 7th Annual Bridging the Spectrum Symposium, Washington, D.C.
- Yontz, E., Hsieh-Yee, I., & Houston, S. (2015, February). *Healthy Heroes Summer Reading Club: Developing healthy youth at public libraries*. 11th Annual Jean Mills Health Symposium, Greenville, North Carolina.
- Yontz, E., Hsieh-Yee, I., and Houston, S. (2015, January). *Healthy youth and libraries: A pilot study*. Association for Library & Information Science Education (ALISE) Annual Conference, Chicago, Illinois.

- Hsieh-Yee, I. (2014, May). *Linking CRRRA resources to portal themes via authority files*. Presented at the Catholic Research Resources Alliance 2014 Membership Meeting, Marquette, WI.
- Hsieh-Yee, I. (2014, April). *Enhancing subject access to CRRRA resources*. Presented at the 2014 Catholic Library Association Conference, Pittsburgh, PA.
- Hsieh-Yee, I. (2014, January). *Health Information Technology Program: Educational entrepreneurship in action*. Presented at the 2014 annual Conference of the Association for Library and Information Science Education, Philadelphia, PA
- Hsieh-Yee, I., Zhang, S., Lin, K., and Cherry, S. (2014, January). *Discovering information through Summon: An analysis of user search strategies and search success*. Paper presented at the 6th Bridging the Spectrum Symposium, Washington, D.C.
- Hsieh-Yee, I. (2012, December). *National Digital Stewardship Alliance and SLIS at CUA: An Educational Partnership*. Paper presented at Best Practices Exchange: Acquiring, Preserving, and Providing Access to Government Information in the Digital Era, Annapolis, MD
- Choi, Y. and Hsieh-Yee, I. (2010, January). *Finding Images in an OPAC: Analysis of User Queries, Subject Headings, and Description Notes*. Paper presented at 2nd Annual Bridging the Spectrum Symposium, Catholic University of America, Washington, D.C.
- Hsieh-Yee, I. and Coogan, J. (2010, January). *Google Scholar vs. Academic Search Premier: What Libraries and Searchers Need to Know*. Paper presented at 2nd Annual Bridging the Spectrum Symposium, Catholic University of America, Washington, D.C.
- Hsieh-Yee, I. (2009, November). *Information Science Education: An LIS School's Perspective*. Paper presented at Annual Meeting of the American Society for Information Science and Technology, Vancouver, British Columbia, Canada.
- Hsieh-Yee, I., Menard, E., Ya-Ning Chen, A., Shu-Jiun Chen, S., Kalfatovic, M. R., Wisser, K. M. (2009, November). *Information Organization in Libraries, Archives and Museums: Converging Practices and Collaboration Opportunities*. Presented at Annual Meeting of the American Society for Information Science and Technology, Vancouver, British Columbia, Canada. (Organizer and moderator of this panel.)
- Hsieh-Yee, I. and Coogan, J. (2009, July). *Catching up to Google Scholar: The Retrieval Power of Academic Search Premier and Google Scholar*. Poster presented at American Library Association Conference, Chicago, Illinois.
- Hsieh-Yee, I., with the CUA Scholarly Communications Project Team. (2009, January). *Digital Scholarship@CUA: Developing an Institutional Repository for CUA*. Poster presented at

1st Annual Bridging the Spectrum Symposium, Catholic University of America, Washington, D.C.

Wise, M., Cylke, K., and Hsieh-Yee, I. (2009, January). *Digital Talking Books: Meeting the Needs of the Blind and the Handicapped*. Paper presented at the Bridging the Spectrum Symposium, Catholic University of America, Washington, D.C.

Hsieh-Yee, I. (2009, January). *User Expectations of MERIC*. Presented at the Information Organization Competencies for the 21st Century Discussion Session of the 2009 Conference of the Association for Library and Information Science Education, Denver, Colorado.

Choi, Y., and Hsieh-Yee, I. (2008, November). *Subject Access for Images in an OPAC*. Annual Meeting of the American Society for Information Science and Technology, Columbus, Ohio. (Also co-organized a panel on Retrieving and Using Visual Resources: Challenges and Opportunities for Research and Education.)

Hsieh-Yee, I. (2008, June). *Educating Cataloging Professionals in a Changing Information Environment*. National Taiwan University, Taipei, Taiwan.

Vellucci, S. L., Moen, W.E., Hsieh-Yee, I., Marson, B., and Wisser, K. (2008, January) *Building a Metadata Education and Research Community through MERIC (Metadata Education and Research Information Commons): Demo and Stakeholder Input*. A panel presented at the 2008 Conference of the Association for Library and Information Science Education, Philadelphia, Pennsylvania.

Hsieh-Yee, I., Choi, Y. and Kules, B. (2007, October). *Searching for Books and Images in OPAC: Effects of LCSH, TOC and Subject Domains*. A poster presented at the American Society for Information Science and Technology Annual Meeting, Milwaukee, Wisconsin.

Hsieh-Yee, I. and Coogan, J. (2007, August) *Google Scholar vs. Academic Search Premier: A Comparative Analysis*. Presented to the Faculty and Staff of the University of the District of Columbia.

Hsieh-Yee, I. and Coogan, J. (2007, June). *Google Scholar vs. Academic Search Premier: A Comparative Analysis*. Presented to the Washington Research Library Consortium Community, Catholic University of America, Washington, D.C.

Hsieh-Yee, I., Choi, Y., and Kules, B.. (2007, June). *What Users Need for Subject Access: Table of Contents or Subject Headings?* A poster presented at the 2007 American Library Association Annual Conference, Washington, D.C., June 2007.

Choi, Y., Hsieh-Yee, I., and Kules, B. (2007, June). *Retrieval Effectiveness of TOC and LCSH*. A paper presented at the Joint Conference on Digital Libraries 2007, Vancouver, Canada.

- Vellucci, S. L., Hsieh-Yee, I., and Moen, W.E. (2007, May). *If We Build It, Will They Come? Building a Community of Practice for Metadata Stakeholders*. A poster presented at the Rutgers University Research Day, Bridgeton, New Jersey.
- Hsieh-Yee, I. (2007, May). *Federated Searching: User Experience & Perceptions*. International Conference on Information Organization & Retrieval, National Taiwan University, Taipei, Taiwan.
- Hsieh-Yee, I. (2007, May). *Search Performance of Google Scholar and Academic Search Premier*. International Conference on Information Organization & Retrieval, National Taiwan University, Taipei, Taiwan.
- Hsieh-Yee, I. (2007, May) *MERIC: Building a Digital Commons for Metadata Education & Research*. International Conference on Information Organization & Retrieval, National Taiwan University, Taipei, Taiwan.
- Hsieh-Yee, I., and Coogan, J. (2007, March/April). *A Comparative Analysis of Google Scholar and Academic Search Premier*. Poster presented at the Association of College & Research Libraries 13th National Conference, Baltimore, Maryland.
- Vellucci, S. L. and Hsieh-Yee, I. (2007, March/April) *They Didn't Teach Me That in Library School! Building a Digital Teaching Commons to Enhance Metadata Teaching, Learning and Research*. On-site presentation and Webcast by Elluminate. A contributed paper presented at the Association of College & Research Libraries 13th National Conference, Baltimore, Maryland. The acceptance rate for contributed paper was 20%. This paper was one of 10 conference papers chosen for live webcast during the conference.
- Moen, W., Hsieh-Yee, I. and Vellucci, S.L. (2007, January) *A DSpace Foundation for a Teaching & Research Commons: The Metadata Education and Research Information Commons*. A poster session presented at the Open Repositories Conference 2007, San Antonio, Texas.
- Tang, R., Hsieh-Yee, I., and Zhang, S. (2006, November) *User Perception of MetaLib Combined Search*. Paper presented at the Annual Meeting of the American Society for Information Science and Technology, Austin, Texas, Nov. 2006.
- Hsieh-Yee, I. (2006, November). *Federated Searching: User Perceptions, System Design, and Library Instructions*. Paper presented at the Annual Meeting of the American Society for Information Science and Technology, Austin, Texas. (Panel organizer, moderator, presenter).
- Hsieh-Yee, I. (2006, November). *Building a Digital Teaching Commons to Enhance Teaching and Learning: The MERIC Experience and Challenges*. Paper presented at the Annual Meeting of the American Society for Information Science and Technology, Austin,

Texas. (Panel organizer, moderator, presenter)

Hsieh-Yee, I. (2006, September). *Search Performance of Google Scholar and Academic Search Premier*. Paper presented at the ERIC Publishers Meeting, Washington, D.C.

Hsieh-Yee, I., Zhang, S., and Rong Tang, R. (2006, June). *User Perceptions of a Federated Search System*. Poster presented at Joint Conference on Digital Libraries, Chapel Hill, North Carolina.

Hsieh-Yee, I. and Zhang, S. (2006, June). *Preparing Users for Federated Search: Implications of a MetaLib User Perceptions Study*. Paper presented at the 2006 Ex Libris User Groups of North America Conference, Knoxville, Tennessee.

Hsieh-Yee, I. (2006, January). *MERIC Organizations and Navigation*. Paper presented at the 2006 ALISE Annual Conference, San Antonio, Texas.

Hsieh-Yee, I. (2006, January). *Metadata and Cataloging Education: Recommended Competencies*. Paper presented at the 2006 ALISE Annual Conference, San Antonio, Texas.

Hsieh-Yee, I. (2005, November). *Digital Library Evaluation: Progress & Next Steps*. Presentation at the Annual Meeting of the American Society for Information Science & Technology, Charlotte, North Carolina.

Hsieh-Yee, I. (2005, August). *Providing Access to Digital Content: Issues for DL Managers*. Presentation at MDK12 Digital Library Steering Committee Meeting, Columbia, Maryland.

Hsieh-Yee, I. (2005, April). *Enhancing Teaching and Learning: The Role of School Library Media Specialists*. Presentation at Meeting of the Baltimore County Public School System School Media Specialists, Baltimore, Maryland.

Hsieh-Yee, I. (2005, January). *Subject Access and Users: Insights & Inspirations from Marcia J. Bates*. Paper presented at the Historical Perspectives SIG, 2005 Conference of the Association for Library and Information Science Education, Boston, Massachusetts.

Hsieh-Yee, I. (2005, January). *Electronic Resource Management: Practice, Employer Expectations, & CE Interests*. Paper presented at Technical Services Education SIG, 2005 Conference of the Association for Library and Information Science Education, Boston, Massachusetts.

Hsieh-Yee, I. (2004, October). *Library Professionals for the Digital Age: Competencies & Preparation*. Paper presented at Bibliographic Access Management Team meeting, Library of Congress, Washington, D.C.

- Hsieh-Yee, I. (2004, January). *Cataloging and metadata expertise for the digital era*. Presented at Preparing 21st Century Cataloging and Metadata Professionals: A Workshop for Educators and Trainers, San Diego and sponsored by ALCTS, ALISE, LC, and OCLC.
- Hsieh-Yee, I. (2004, January). *Educating catalogers for the digital era*. Paper presented at the Technical Services SIG, 2004 Conference of the Association for Library and Information Science Education, San Diego.
- Hsieh-Yee, I. (2003, July). *Cataloging Education for the 21st Century*. A presentation at the Library of Congress, Washington, D.C.
- Hsieh-Yee, I. (2002, January) *Metadata Education and Research Priorities: A Delphi Study of Metadata Experts*. Presentation at the 2002 Conference of the Association for Library and Information Science Education, New Orleans.
- Hsieh-Yee, I. (2001, November). *A Delphi Study of Metadata: Preliminary Findings*. Poster session at the 2001 Annual Meeting of the American Society for Information Science & Technology, Washington, D.C.
- Hsieh-Yee, I. (2001, June). *Resources on Asian American Children: Analysis of Retrieval by Search Engines and WorldCat*. Presentation at the National Conference on Asian Pacific American Librarians, San Francisco.
- Hsieh-Yee, I. (2001, January). *Delphi Study on Metadata: Project Design*. Presentation at Research Awards Session, Association for Library & Information Science Education, Washington, D.C.
- Hsieh-Yee, I. (2000, May). *Web Search Behavior Research: Progress and Implications*. Presentation at the Symposium on Evaluating Library and Information Science Research, University of Wisconsin-Madison, Madison, Wisconsin.
- Hsieh-Yee, I. (2000, March). *ERIC User Services: Evaluation in a Decentralized Environment*. Presentation at the National ERIC Joint Directors/Technical Meeting, Arlington, Virginia.
- Hsieh-Yee, I. (2000, January). *Enhancing Learning with Web Technology*. Presentation at Faculty Conversations, Catholic University of America, Washington, D.C.
- Hsieh-Yee, I. (2000, January). *From Surrogates to Objects: CUA's Approaches to Organizing Electronic Resources*. Paper presentation at the Annual Conference of the Association for Library and Information Science Education, San Antonio, Texas.
- Yee, P., and Hsieh-Yee, I. (1997, November). *Individual Differences in Search Behavior on the WWW*. A poster session presented at the 38th Annual Meeting of the Psychonomic

Society, Philadelphia, Pennsylvania.

Hsieh-Yee, I. (1997, April). *Research + Marketing + Preparation = Job!* Presented at the "Workshop on Resume and Interview Techniques," Special Libraries Association, Student Chapter, Catholic University of America, Washington, D.C.

Hsieh-Yee, I. (1997, February). *Creating CyberCatalogers: Education and Training.* Presentation at ALA's Midwinter Meeting, Washington, D.C.

Hsieh-Yee, I. (1997, February). *Search Tactics of Web Users in Searching for Texts, Graphics, Known Items and Subjects: A Search Simulation Study.* Presented at the Conference of the Association for Library and Information Science Education, Washington, D.C.

Hsieh-Yee, Ingrid. "Beginning Your Special Library/Information Center Career." Presented at SLA's "Career Day," Jan. 11, 1997, Catholic University of America.

Hsieh-Yee, I. (1996, September). *The Roles of Library and Information Scientists in Managing Electronic Information.* Presentation at Hamilton College, Clinton, New York.

Hsieh-Yee, I. (1996, May). *The Future of Cataloging as a Profession.* Presented at "The Cataloging Forum, Library of Congress, Washington, D.C.

Hsieh-Yee, I. (1994, October). *The Impact of the Internet on OPACs.* Presented at the Third Workshop on User Interfaces for OPACs, Library of Congress, Washington, D.C.

Reports

Hsieh-Yee, I., with Knowledge Management Competencies and Performance Action Group of the Federal Knowledge Management Initiative. "From Knowledge Management Competencies to Improved Organizational Performance." April 9, 2009.

Hsieh-Yee, I., with Knowledge Practices Action Group of the Federal Knowledge Management Initiative. "KM Practice in Government Agencies: Findings and Recommendations." April 9, 2009.

Hsieh-Yee, I. "Delphi Study on Metadata." 2001. Three quarterly reports submitted to the Association for Library and Information Science Education.

Hsieh-Yee, I. "College Students' Information Channels: Patterns of Use and Possible Factors in Channel Selection." 1995. Submitted to the Catholic University of America.

Hsieh-Yee, I. "The Information-Seeking Patterns of Scholars and Their Use of an Online Information System." 1994. Submitted to the Council on Library Resources.

Book Reviews

- Review of *The Measurement and Evaluation of Library Services*, by Sharon L. Baker and F. Wilfrid Lancaster. *Information Processing and Management* 30 (1994): 450-52.
- Review of *Subject Access to Films and Videos*, by Sheila S. Intner and William E. Studwell; and *Cataloging Unpublished Nonprint Materials*, by Verna Urbanski with Bao Chu Chang and Bernard L. Karon. *Information Processing and Management* 30 (1994): 449-50.
- Review of *Automated Information Retrieval in Libraries: A Management Handbook*, by Vicki Anders. *Journal of Library and Information Science* 19 (1993): 98-100.
- Review of *Full Text Databases*, by Carol Tenopir and Jung Soon Ro. *Information Processing and Management* 28 (1992): 667-68.
- Review of *Descriptive Cataloging for the AACR2R And USMARC: A How-to-Do It Workbook*, by Larry Millsap and Terry Ellen Ferl. *Information Processing and Management* 28 (1992): 809-11.
- Review of *MARC Manual: Understanding and Using MARC Records*, by Deborah J. Byrne. *Information Processing and Management* 28 (1992): 537-38.

Service

Professional Associations and Societies

- Library of Congress. RDA Training Program for the Profession. Co-authored with Tim Carlton. 2013-2014.
- 2014 Digital Preservation Outreach & Education Survey. Contributed to the design of the survey, 2014.
- National Digital Stewardship Alliance. Outreach Committee. 2011-2014.
- National Digital Stewardship Residency Program. Advisory Group, 2012-2013.
- FEDLINK Health Information Technology Advisory Council, 2011-2015.
- 2012 Joint Conference on Digital Libraries. Program Planning Committee, Pre-Conference Proposals Review Committee, 2012
- Catholic Research Resources Alliance. Five-Year Strategic Plan Task Force, 2011-2012
- Institute of Museum and Library Services. Grant reviewer. 2004, 2005, 2010.
- Association for Library and Information Science Education.
 - * ALISE Bodan Wynar Research Paper Award Committee, 2015, 2016, 2017
 - * ALISE Eugene Garfield Dissertation Award Competition, Jury, 2013, 2014
 - * ALISE Research Grant Competition Committee. Chair, 2012
 - * Pratt-Severn Faculty Innovation Award. Chair, 2009, 2010
 - * ALISE Doctoral Poster Jury, 2012
 - * "Information Organization Competencies for the 21st Century" Discussion session leader. 2009 Conference of the Association for Library and Information Science

Education.

* Assisted Technical Services SIG Convener in organizing a program, ““Building a Metadata Education and Research Community through MERIC (Metadata Education and Research Information Commons): Demo and Stakeholder Input” for the 2008 ALISE conference.

* Association for Library Collections and Technical Services/Association for Library and Information Science Education (ALCTS/ALISE) Metadata Education and Research Information Center (MERIC) Advisory Board, Co-Chair (with Sherry Vellucci), 2005-2007. Chair, 2008-2009 (leading the effort to build MERIC, a repository and collaborative space for metadata educators, practitioners, and researchers)

* Technical Services SIG, Convener, 2004-2005. Organized a program on “Electronic Resources Management: Current Practices, Employer Expectations, and Teaching Strategies” for the 2005 conference in Boston, Massachusetts.

* Technical Services SIG, Convener, 2003-2004. Organized a program on “Organizing Information with Metadata: Desired Competencies and Teaching Innovations” for the 2004 conference.

* Technical Services SIG, Convener, 1999-2000. Organized a program on "Teaching the Organization of Electronic Resources" for the 2000 conference.

* Curriculum SIG, Co-convener (with Sibyl Moses), 1996-97. Organized a program on “Government Information Policy” for the 1997 conference.

- American Society for Information Science & Technology.

* Reviewer, Conference program panel submissions and poster submissions, 2005, 2006, 2007, 2009, 2011, 2012, 2013, 2014, 2015, 2016, 2017

* Nomination Committee, 2009-2011

* Information Science Education Special Interest Group. American Society for Information Science and Technology. Chair-Elect, 2007-2008. Chair 2008-2009.

* Committee on Information Science Education. 1999-2006.

* Committee on Information Science Education. Organizing Committee for an orientation program for students at ASIS annual meetings, 1999-2001

* Committee on Information Science Education. Sub-committee on Student Welfare (focusing on issues related to master's education), 1998-2001

* SIG ED. Organizing Committee for the "Seminar on Research and Career Development" for junior researchers. 1995-96 (chair), 1997-2001

* ISI Doctoral Dissertation Proposal Scholarship Jury, 1997; 2001, 2002

* Pratt-Severn Best Student Research Paper Award Jury. Chair. 1997

* 1998 Midyear Meeting (referee of contributed papers), 1997

* Organizer and moderator of the ASIS Doctoral Forum and the Doctoral Research Seminar 1994-1995

* SIG Human Computer Interaction. Chair-Elect, Chair, 1993-1995

* Doctoral Forum Award Jury, 1995

* Best Student Paper Award Jury, 1995

- American Library Association.

- * Committee on Accreditation, External Review Panelist, 2009- (site visiting team 2013-2014; site visiting team 2016-2017)
- * Association for Library Collections and Technical Services Task Force on Competencies and Education for a Career in Cataloging, member, 2008-2009
- * Facilitator for "What They Don't Teach in Library School: Competencies, Education and Employer Expectations for a Career in Cataloging," an Association for Library Collections and Technical Services Preconference, June 22, 2007 in Washington, D.C. Also a local liaison for bringing this program to the Catholic University of America.
- * Facilitator for a discussion on "Effect of Electronic Resources on Technical Services" at ALA's Midwinter Meeting held in Feb. 1997 in Washington, D.C.
- * International Relations Committee, Subcommittee Task Force for IFLA and China, 1994-1997
- Virginia Association of School Librarians. Scholarships and Awards Committee. 2010-2012
- Federal Knowledge Management Initiative, Knowledge Management Practices Action Group. Member. 2009 (leading the effort to build a knowledge management repository)
- Federal Knowledge Management Initiative, Knowledge Management Competencies & Learning Action Group. Member. 2009 (developing an action plan for helping government knowledge workers and government agencies to develop knowledge management competencies)
- National Center for Education Statistics. Technical Review Panel. 2008.
- External evaluator for a case of promotion to full professorship. University of Tennessee. 2008.
- National Information Standards Organization (NISO). Advisory Board, Revision of "IMLS Framework of Guidance for Building Good Digital Collections," 2004, 2007.
- Library of Congress, Bibliographic Control of Web Resources: A Library of Congress Action Plan. Principal Investigator of Action Item 5.1, focusing on cataloging and metadata education for students and new librarians, 2002-2003. (worked with the Association for Library Collections and Technical Services, Education Task Force)
- Chinese American Librarians Association
 - * Chinese American Librarians Association Outstanding Library Leadership Award in Memory of Dr. Margaret Chang Fung, Award Committee, 2016-2017
 - * Achievement Award Jury, 2000-2001
 - * CALA Goal 2000 Task Force, 1997
 - * Scholarship Committee, 1995, 1996-1997 (chair)
 - * Board of Directors, 1994-1997
 - * Publication Committee, 1993-1995
 - * International Relations Committee, 1993-1996
- SailorSM Assessment Advisory Group (An impact study of Sailor, Maryland's Public Information Network), 1995

- Editorial boards
 - Journal of Library & Information Science. Editorial Board, 2012-
 - Chinese American Librarians Association, *Occasional Papers Series*. Editorial Board, 2009-2016.
 - Library Quarterly*. Editorial Board, 2003-2008
 - Bulletin of the Medical Library Association*, 1994-97
 - Newsletter editor for the Chinese American Librarians Association, 1989-92
- Referee for the following journals
 - Information Processing and Management*
 - Journal of Digital Information*
 - Journal of Education for Library and Information Science*
 - Journal of Information Science*
 - Journal of Library & Information Science*
 - Journal of Library Metadata*
 - Journal of the American Society for Information Science & Technology*
 - Library and Information Science Research*
 - Library Quarterly*
- Expert reviewer, “Digital Library” course, Evaluation module, University of North Carolina, Chapel Hill, 2007-2008.
- Expert reviewer, “Information Organization” course, University of Michigan, Ann Arbor. 2007.

Catholic University of America

- Academic Senate Committee on Appointments and Promotions, 2005-2008
- Academic Senate Committee on Committees and Rules, 2009-2012, 2019-2022
- Academic Senate representative (for School of Arts & Sciences), 2017-2020
- Academic Senate representative (for School of Library & Information Science) 2003-2012
- Academic Senate Library Committee, Interim Chair (2007), Member, 2008-2012
- Academic Senate Committee on Computing, 1995-2003
- President’s Administrative Council, 2010-2012
- Deans’ Council, 2010-2012
- Academic Leadership Group, 2010-2012
- Graduate Board, 2010-2012
- School of Arts & Sciences, Committee on Appointments and Promotions, 2015-2019
- School of Arts & Sciences, Academic Council, 2015-2016.

- School of Arts & Sciences, Ordinary Professor Group, 2013-
- Doctoral Dissertation Defense Committee, Chair, Dept. of Psychology, 2016, 2017, 2018, 2019, 2020, 2021, 2022
- Doctoral Dissertation Defense Committee, Chair, Dept. of Education, 2014, 2015, 2017, 2018, 2019
- Doctoral Dissertation Defense Committee, Chair, National Catholic School of Social Services, 2019, 2020, 2021
- Doctoral Dissertation Defense Committee, Chair, School of Nursing, 2006, 2008
- CUA Scholarly Communication Project Team, Member (2007), Chair, 2008-2009
- Dean Search Committee, 1992-1994, 1998-1999, 2002-2003, 2006-2007
- Fulbright Review Panel, 2006
- CUA Service Learning Advisory Board, 2001-2002
- CUA Faculty Conversations on Enhancing Teaching and Learning through Technology, Planning Group, 1999-2001
- CUA Initiative on Technology and Teaching, 1998-2001

Dept. of Library and Information Science

- Admissions Committee, 2007-2009, Chair 2010-2012, Member 2013-2015, Member 2018-2022
- Appointments and Promotions Committee, 1991-2022
- Student advisement, 1990-2022
- 4+1 Bachelor to Master program, Program Advisor, 2014-2022
- LIS Minor, Advisor, 2017-2022
- Curriculum Committee, 2020-21 (Member), 2021-2022 (Chair)
- Scholarship & Awards Committee, 2016-2018, 2020-21
- Comprehensive examination editor, 2020-21
- Symposium, Colloquium, Lecture Series Committee, May 2018-May 2020 (Chair)
- Symposium and Colloquium Committee, fall 2016-May 2018
- Bridging the Spectrum Symposium session moderator, 2018, 2019, 2020, 2021
- Organization of Information Core Course Outcomes Assessment, 2019, 2020
- The Information Professions in Society Core Course Outcomes Assessment, 2019, 2020
- Community Services Librarianship, Course of Study Review and Revision, 2020
- Organization of Information, Course of Study Review and Revision, 2019
- Organization of Information, Competency and Course Offerings Review, 2018-2019
- Accreditation presentation, Chair, June 2015-August 2016
- Interim Co-Chair, June 2015-August 2016.
- Blended/OWL Learning Committee, spring 2016-2018
- Technology Committee, fall 2016-2017
- Comprehensive examination editor, 2016-2017, reader (every year since 1990)
- LIS Advisory Board, 2015-2016 (chair); fall 2016- May 2018 (member)
- Committee on Planning and Assessment, 2015-2016 (chair)

- Senior Faculty Committee, 2014-2016.
- Accreditation Steering Committee, 2014-2016 (Chair, 2015-2016)
- Accreditation Students Standard Committee, co-chair, 2014-2016
- Accreditation Mission, Goals, and Objectives Standard Committee, co-chair, 2014-2016
- Accreditation Curriculum Standard, member 2014-2-16
- Accreditation Administration and Finance Standard, member 2014-2016
- Cultural Heritage Information Management Project (IMLS-funded), Co-PI, 2012-2015
- Cultural Heritage Information Management Forum (scheduled for June 2015), Co-Organizer, 2013-2015
- Health Information Technology Interim Review Committee, 2015 (chair)
- Health Sciences Librarianship Advisory Group, 2015- (chair)
- Comprehensive examination editors, 2013-2014, 2016-2017
- National Digital Stewardship Alliance liaison, 2011-2014
- Advisory Board, Chair 2010-2012
- Academic Honesty Committee, Chair, 2008-2012
- Blended Learning Committee, 2010-2012
- Colloquium Committee, 2010-2012
- Comprehensive Examination Administration, 2010-2012
- Cultural Heritage Information Management Advisory Committee, 2010-2012 (chair), 2013-
- Curriculum Committee, 1991-2003, 2007-2009, Chair 2010-2012, member 2013-
- Curriculum Subcommittee on Comprehensive Examination, Chair 2009-2012
- Health Information Technology Advisory Board, Chair 2010-2012. Member 2013-
- Health Sciences Advisory Committee, 2009, Chair 2010-2012. Member 2013-
- HIT Expert Forum, Chair 2012. Member 2013-2015
- Health Information Technology Student Group Advisor, 2011-2012
- State Council for Higher Education of Virginia, SLIS Representative, 2010-2012
- Symposium Planning Committee, 2010-2012
- Website Management Team, Chair, 2010-2012
- Urban School Librarianship Project (IMLS-Funded), PI, 2007-2011 (chair, 2010-11)
- Failing Grades Committee, 1995-1997 (chair), 2000-2001 (chair), 2004-2005 (chair), 2007 (chair)-2011
- Faculty Search Committee, 1994-1998, 2002-2004, 2006 (chair), Fall 2007-2009, Chair fall 2009-2012
- Recruitment Committee, Chair 2010-2012
- Strategic Planning Committee, Chair 2010-2012
- Technology Committee, 2010-2012
- Accreditation Advisory Committee, 2007-2009
- Accreditation Coordinating Committee, 2007-2009
- Accreditation Steering Committee, 2007-2009
- SLIS Advisory Group, 2007-2009
- Accreditation Curriculum Standard Committee, Co-chair, 2007-2009
- Accreditation Faculty Standard Committee, Co-chair, 2007-2009

- LSC 551 Information Organization Review Team, Co-chair, 2008-2009, 2015-2016.
- Curriculum Subcommittee on Portfolios, 2009
- LSC 555 Information Systems in Libraries and Information Centers Review Team, contributor, 2008-2009
- Redesign of LSC 730 Use and Users of Libraries and Information. 2009-
- Development of a metadata institute that was taught as LSC 715 Organization of Internet Resources in 2008. The institute is being revised and will be offered in 2010 under a new course title.
- Development of lesson plans, assignments, and evaluation rubrics for LSC 606, Cataloging and Classification, for the School's NCATE accreditation. 2008
- Howard and Mathilde Rovelstad Scholarship Committee, Chair, 2004-2007
- Assistant Dean Search Committee, Chair, Fall 2007
- Liaison to the Association for Library Collections and Technical Services to bring its preconference program, Cataloging Education and Employer Expectations, to CUA during the 2007 American Library Association Annual Meeting in Washington, D.C.
- Organizer of the colloquium presentation and reception for Tamar Sadeh of Ex Libris on PRIMO June 2007
- Practicum review and design (work with potential supervisors, such as the American Indian Museum internship description revision) 2006-
- Comprehensive examinations (edits, proctoring, and grading), 1990-
- SLIS Web site redesign: Comments and suggestions. Fall 2007
- Conducted surveys of current students and alumni in preparation for the 2005 re-accreditation, 2004-2005
- Technology Committee, 1992-1999 (chair, 1996-1998), 2002-2003 (member)
- Colloquia Committee 1997-1999, 2002-2003.
- Advisor of the CUA Student Chapter of the American Society for Information Science and Technology, 2002-2003
- Visiting Professor Search Committee, 1999, 2000, 2001
- Leader, Participation in the CORC experiment, 1999-2000
- Advisor of the Special Libraries Association Student Chapter, 1993-1999; the group was recognized for outstanding leadership by SLA in 1999.
- COA planning Committee, Task Force on Electronic Presentation of SLIS Reports (team leader) 1997-1998
- COA Planning Committee, Subcommittee on Technology 1996-1998
- NLM practicum coordinator, 1997-1998
- Computer Literacy Workshops: Assisted with the development and evaluation of the workshops, 1996-1998
- Leader, Participation in the InterCat project, 1995-1997

APPENDIX B

[Library of Congress](#) >> [MARC](#) >> [Understanding MARC](#)

MARC 21 Reference Materials

[Part VII: A Summary of Commonly Used MARC 21 Fields](#)

[Part VIII: A List of Other Fields Often Seen in MARC Records](#)

[Part IX: The Leader](#)

[Part X: Field 008 for Books](#)

Part VII:

A Summary of Commonly Used MARC 21 Fields

This is a summary of the MARC 21 tags used most frequently by libraries in entering their own bibliographic records. For full listings of all MARC 21 tags, indicators, and subfield codes, see *MARC 21 Format for Bibliographic Data*.

In the explanations on these pages:

Tags -- The tags (3-digit numbers) are followed by the names of the fields they represent. In this summary, and in the *MARC 21 Format for Bibliographic Data*, if a tag can appear more than once in one bibliographic record, it is labeled repeatable (R). If it can only be used once, it is labeled non-repeatable (NR). For example, a catalog record can have several subjects, so the tags for subject added entries (6XX) are labeled repeatable (R).

Indicators -- The use of indicators is explained in fields where they are used. Indicators are one-digit numbers. Beginning with the 010 field, in every field -- following the tag -- are two character positions, one for Indicator 1 and one for Indicator 2. The indicators are not actually defined in all fields, however. And it is possible that a 2nd indicator will be used, while the 1st indicator remains undefined (or vice versa). When an indicator is undefined, the character position will be represented by the character # (for blank space).

Subfield codes -- All the data in each field (beginning with the 010 field) is divided into subfields, each of which is preceded by a delimiter-subfield code combination. The most common subfield codes used with each tag are shown. Each subfield code is preceded by the character \$, signifying a delimiter. The name of the subfield follows the code.

In general, every field MUST have a subfield 'a' (\$a). One exception that is often seen is in Field 020 (ISBN), when the ISBN information (subfield \$a) is unavailable but the price (subfield \$c) is known. Some subfields are repeatable. In this summary, repeatability is noted for only the more common repeatable subfields.

Examples: Examples follow the explanation for each field. For clarity, one space has been placed between the tag and the first indicator, one space has been placed between the second indicator and the first delimiter- subfield code, and one space has been inserted between the delimiter-subfield code and the subfield data.

010 Library of Congress Control Number -- (LCCN)

(NR, or Not Repeatable)

*Indicators undefined.**Subfield used most often:***\$a** -- Library of Congress control number

Example: 010 ## \$a ###86000988#

020 International Standard Book Number -- (ISBN)

(R, or Repeatable)

*Indicators undefined.**Subfields used most often:***\$a** -- International Standard Book Number**\$c** -- Terms of availability (often a price)**\$z** -- Cancelled/invalid ISBN (R)

Example: 020 ## \$a 0877547637

040 Cataloging source -- (NR)*Indicators undefined.**Subfields used most often:***\$a** -- Original cataloging agency**\$c** -- Transcribing agency**\$d** -- Modifying agency (R)

Example: 040 ## \$a DLC
 \$c DLC
 \$d gwhs

100 Main entry -- Personal name -- (primary author)

(NR; there can be only one main entry)

Indicator 1: Type of personal name entry element

0 -- Forename

1 -- Surname (this is the most common form)

3 -- Family name

Indicator 2 undefined.

Indicator 2 became obsolete in 1990. Older records may display 0 or 1

*Subfields used most often:***\$a** -- Personal name**\$b** -- Numeration

\$c -- Titles and other words associated with a name (R)
\$q -- Fuller form of name
\$d -- Dates associated with a name (generally, year of birth)

Example: 100 1# \$a Gregory, Ruth W.
 \$q (Ruth Wilhelme),
 \$d 1910-

130 Main entry -- Uniform title -- (NR)

Indicator 1: Nonfiling characters

0-9 -- Number of nonfiling characters present (for initial articles, including spaces)

Indicator 2 undefined.

Indicator 2 became obsolete in 1990. (See 100 above.)

Subfields used most often:

\$a -- Uniform title
\$p -- Name of part/section of a work (R)
\$l -- Language of a work
\$s -- Version
\$f -- Date of a work

Example: 130 0# \$a Bible.
 \$p O.T.
 \$p Psalms.

240 Uniform title (NR)

Indicator 1: Uniform title printed or displayed

0 -- Not printed or displayed

1 -- Printed or displayed (most common)

Indicator 2: Nonfiling characters

0-9 -- Number of nonfiling characters present (for initial articles, including spaces)

Subfields used most often:

\$a -- Uniform title
\$l -- Language of a work
\$f -- Date of a work

Example: 240 10 \$a Ile mystérieuse.
 \$l English.
 \$f 1978

245 Title Statement (NR)

Indicator 1: Title added entry

(Should the title be indexed as a title added entry?)

0 -- No title added entry

(indicates a title main entry; i.e. no author is given)

1 -- Title added entry

(the proper indicator when an author given in 1XX; the most common situation)

Indicator 2: Nonfiling characters

0-9 -- Number of nonfiling characters present, including spaces; usually set at zero, except when the title begins with an article; e.g., for *The robe*, the second indicator would be set to 4. The letters *T*, *h*, *e*, and the space following them are then ignored in alphabetizing titles. The record will be automatically filed under "r" -- for *Robe*.

Subfields used most often:

\$a -- Title proper

\$h -- Medium (often used for non-book media)

\$p -- Name of part/section of a work (R)

\$b -- Reminder of title (subtitles, etc.)

\$c -- Remainder of title page transcription/Statement of responsibility

Example: 245 14 \$a The DNA story :
 \$b a documentary history of gene
 cloning /
 \$c James D. Watson, John Tooze.

246 Varying form of title (R)

Indicator 1: Note/title added entry controller

1 -- Note, title added entry

3 -- No note, title added entry

Indicator 2: Type of title

-- No information provided

0 -- Portion of title

1 -- Parallel title

4 -- Cover title

8 -- Spine title

Subfield used most often:

\$a -- Title proper

Example: 246 3# \$a Four corners power review

250 Edition statement (NR)

Indicators undefined.

Subfield used most often:

\$a -- Edition statement

Example: 250 ## \$a 6th ed.

260 Publication, distribution, etc. (Imprint) (R)

Indicator 1: Sequence of publishing statements

-- No information provided

Indicator 2: *Undefined*

Subfields used most often:

\$a -- Place of publication, distribution, etc. (R)

\$b -- Name of publisher, distributor, etc. (R)

\$c -- Date of publication, distribution, etc. (R)

Example: 260 ## \$a New York :
 \$b Chelsea House,
 \$c 1986.

300 Physical description (R)

Indicators undefined.

Subfields used most often:

\$a -- Extent (number of pages) (R)

\$b -- Other physical details (usually illustration information)

\$c -- Dimensions (cm.) (R)

\$e -- Accompanying material (for example, "teacher's guide" or "manual")

Example: 300 ## \$a 139 p. :
 \$b ill. ;
 \$c 24 cm.

440 Series statement / Added entry--Title

This field was made obsolete in 2008 to simplify the series statement. See 490 and 830.

490 Series statement (No added entry is traced from field) (R)

Indicator 1: Specifies whether series is traced (whether an 8XX tag is also present)

0 -- Series not traced

1 -- Series traced (8XX is in record)

Indicator 2 undefined.

Subfield used most often:

\$a -- Series statement (R)

\$v -- Volume number (R)

Example: 490 1# \$a Colonial American craftsmen

500 General note (R)

Indicators undefined.

Subfield used most often:

\$a -- General note (Used when no specialized note field has been defined for the information. Examples: Notes regarding the index; the source of the title; variations in title; descriptions of the nature, form, or scope of the item.)

Example: 500 ## \$a Includes index.

504 Bibliography, etc. note (R)

Indicators undefined.

Subfield used most often:

\$a -- Bibliography, etc. note

Example: 504 ## \$a Includes bibliographical references.

505 Formatted contents note (R)

Indicator 1: Type of contents note

0 -- Complete contents

1 -- Incomplete contents (used with multivolume set when some volumes are not yet published)

2 -- Partial contents

Indicator 2: Level of content designation

-- Basic

Subfield used most often:

\$a -- Formatted contents note

Example: 505 0# \$a Pride and prejudice -- Emma
-- Northanger Abbey.

520 Summary, etc. note (R)

Indicator 1: Display constant controller

-- Summary

- 1 -- Review
- 2 -- Scope and content
- 3 -- Abstract

Indicator 2 undefined

Subfields used most often

- \$a** -- Summary, abstract, or annotation
- \$b** -- Expansion of summary note

Example: 520 ## \$a This basic guide to parliamentary procedure tells how to conduct and participate in a meeting properly.

600 Subject added entry -- Personal name (R)

Indicator 1: Type of personal name entry element

- 0 -- Forename
- 1 -- Surname (this is the most common form)
- 3 -- Family name

Indicator 2: Subject heading system/thesaurus (identifies the specific list or file which was used)

- 0 -- Library of Congress Subject Headings
- 1 -- LC subject headings for children's literature
- 2 -- Medical Subject Headings
- 3 -- National Agricultural Library subject authority file
- 4 -- Source not specified
- 5 -- Canadian Subject Headings
- 6 -- Répertoire de vedettes-matière
- 7 -- Source specified in subfield \$2

(Note regarding Sears subject headings: The MARC 21 format does not provide an assigned indicator for Sears subject headings. Therefore, an indicator of 7 is used, and the MARC defined code "sears" is placed in subfield \$2.)

Subfields used most often:

- \$a** -- Personal name (surname and forename)
- \$b** -- Numeration
- \$c** -- Titles and other words associated with a name (R)
- \$q** -- Fuller form of name
- \$d** -- Dates associated with a name (generally, year of birth)
- \$t** -- Title of a work
- \$v** -- Form subdivision (R)
- \$x** -- General subdivision (R)
- \$y** -- Chronological subdivision (R)
- \$z** -- Geographic subdivision (R)
- \$2** -- Source of heading or term (used with 2nd indicator of 7)

Example: 600 10 \$a Shakespeare, William,
 \$d 1564-1616
 \$x Comedies
 \$x Stage history.

Example: 600 10 \$a Shakespeare, William,
 \$d 1564-1616
 \$x Knowledge
 \$z Rome
 \$v Congresses.

Notice that subfields \$v, \$x, and \$z in the 600 field are repeatable. Subfields \$v, \$x, \$y, and \$z do not have to be in alphabetical order. They will be in the order prescribed by the instructions given by the subject heading system.

610 Subject added entry -- Corporate name (R)

Indicator 1: Type of corporate name entry element

- 0 -- Inverted name (not used with AACR2)
- 1 -- Jurisdiction name
- 2 -- Name in direct order

Indicator 2: Subject heading system/thesaurus.

See indicator 2 under 600

Subfields used most often:

- \$a -- Corporate name or jurisdiction name as entry element
- \$b -- Subordinate unit (R)
- \$v -- Form subdivision (R)
- \$x -- General subdivision (R)
- \$y -- Chronological subdivision (R)
- \$z -- Geographic subdivision (R)
- \$2 -- Source of heading or term (used with 2nd indicator of 7)

Example: 610 10 \$a United States.
 \$b Army Air Forces
 \$v Biography.

650 Subject added entry -- Topical term (Most subject headings fit here.) (R)

Indicator 1: Level of subject

- # -- No information provided

Indicator 2: Subject heading system/thesaurus

(identifies the specific list or file which was used)

- 0 -- Library of Congress Subject Headings
- 1 -- LC subject headings for children's literature

- 2 -- Medical Subject Headings
- 3 -- National Agricultural Library subject authority file
- 4 -- Source not specified
- 5 -- Canadian Subject Headings
- 6 -- Répertoire de vedettes-matière
- 7 -- Source specified in subfield \$2

Note regarding Sears subject headings: The MARC 21 format does not provide an assigned indicator for Sears subject headings. Therefore, an indicator of 7 is used, and the MARC defined code "sears" is placed in subfield \$2.)

Subfields used most often:

- \$a** -- Topical term
- \$v** -- Form subdivision (R)
- \$x** -- General subdivision (R)
- \$y** -- Chronological subdivision (R)
- \$z** -- Geographic subdivision (R)
- \$2** -- Source of heading or term used with 2nd indicator of 7)

Example: 650 #0 \$a Theater
 \$z United States
 \$v Biography
 \$v Dictionaries.

Notice that subfields \$v, \$x, and \$z in the 650 field are repeatable. Subfields \$v, \$x, \$y, and \$z do not have to be in alphabetical order. They will be in the order prescribed by the instructions given by the subject heading system.

651 Subject added entry -- Geographic name (R)

Indicator 1: undefined.

Indicator 2: Subject heading system/thesaurus.

See indicator 2 under 600

Subfields used most often:

- \$a** -- Geographic name
- \$v** -- Form subdivision (R)
- \$x** -- General subdivision (R)
- \$y** -- Chronological subdivision (R)
- \$z** -- Geographic subdivision (R)
- \$2** -- Source of heading or term (used with 2nd indicator of 7)

Example: 651 #0 \$a United States
 \$x History
 \$v Chronology.

Notice that subfields \$v, \$x, and \$z in the 651 field are repeatable. Subfields \$v, \$x, \$y, and \$z do not have to be in alphabetical order. They will be in the order prescribed by the instructions given by the subject heading system.

700 Added entry -- Personal name (R)

Indicator 1: Type of personal name entry element

- 0 -- Forename
- 1 -- Surname (this is the most common form)
- 3 -- Family name

Indicator 2: Type of added entry

- # -- No information provided (most common; co-authors, editors, etc.)
- 2 -- Analytical entry (The values for Indicator 2 changed in 1994 with Format Integration, and older records may display additional values. An analytical entry involves an author/title of an item contained in a work.)

Subfields used most often:

- \$a** -- Personal name
- \$b** -- Numeration
- \$c** -- Titles and other words associated with a name (R)
- \$q** -- Fuller form of name
- \$d** -- Dates associated with a name (generally, year of birth)
- \$e** -- Relator term (such as ill.) (R)
- \$4** -- Relator code (R)

Example: 700 1# \$a Baldrige, Letitia.

710 Added entry -- Corporate name (R)

Indicator 1: Type of corporate name entry element

- 0 -- Inverted name (not used with AACR2)
- 1 -- Jurisdiction name
- 2 -- Name in direct order

Indicator 2: Type of added entry.

- See Indicator 2 under 700
- # -- No information provided
- 2 -- Analytical entry

Subfields used most often:

- \$a** -- Corporate name or jurisdiction name as entry element
- \$b** -- Subordinate unit (R)

Example: 710 2# \$a Sunburst Communications (Firm)

740 Added entry -- Uncontrolled related/analytical title (R)

Indicator 1: Nonfiling characters

- 0-9 -- Number of nonfiling characters present (for initial articles, including

spaces)

Indicator 2: Type of added entry. See Indicator 2 under 700

-- No information provided

2 -- Analytical entry

(This field was redefined in 1994 with Format Integration. Prior to 1994, the field was also used for variant titles, such as a different wording on a spine title. In records created since Format Integration, those variant titles appear in a 246 field.)

Subfield used most often:

\$a -- Title

Example: 740 02 \$a Uncle Vanya.

800 Series added entry -- Personal name (R)

Indicator 1: Type of personal name entry element

0 -- Forename

1 -- Surname

3 -- Family name

Indicator 2 undefined.

Subfields used most often:

\$a -- Personal name

\$b -- Numeration

\$c -- Titles and other words associated with a name (R)

\$q -- Fuller form of name

\$d -- Dates associated with a name (generally, year of birth)

\$t -- Title of a work (the series)

\$v -- Volume number

Example: 800 1# \$a Fisher, Leonard Everett.
 \$t Colonial American craftsmen.

830 Series added entry -- Uniform title (R)

Indicator 1 undefined.

Indicator 2: Nonfiling characters

0-9 -- Number of nonfiling characters present (for initial articles, including spaces)

Subfield used most often:

\$a -- Uniform title

\$v -- Volume number

Example: 830 #0 \$a Railroads of America (Macmillan)

[\[Back to Top of Page \]](#)

Part VIII:

A List of Other Fields Often Seen in MARC Records

001	Control number
003	Control number identifier
005	Date and time of latest transaction
006	Fixed-length data elements -- additional material characteristics
007	Physical description fixed field
008	Fixed length data elements (See Part X)
022	International Standard Serial Number (ISSN)
037	Source of acquisition
041	Language code
043	Geographic area code
050	Library of Congress call number
060	National Library of Medicine call number
082	Dewey Decimal classification number (the one recommended by the Library of Congress; locally-assigned call numbers may appear elsewhere)
110	Main entry -- Corporate name (less frequent under AACR2 rules)
256	Computer file characteristics
263	Projected publication date (indicates a CIP -- Cataloging in Publication -- record)
306	Playing time
508	Creation/production credits note
510	Citation/references note (review sources)
511	Participant or performer note
521	Target audience note (first indicator: 0 = reading grade level, 1 = interest age level, 2 = interest grade level, 3 = special audience characteristics, 4 = motivation interest level)
530	Additional physical form available note
538	System details note
586	Awards note
656	Index term -- Occupation
730	Added entry -- Uniform title
852	Location
856	Electronic location and access
9XX	Reserved for local use. (They are used by vendors, systems, or individual libraries to exchange additional data)

[\[Back to Top of Page \]](#)

Part IX:

The Leader

There are 24 positions in the Leader, numbered from 00 to 23. For fuller explanation, see the *MARC 21 Format for Bibliographic Data*.

- 00-04 Record length (calculated by the computer for each record)
- 05 Record status
 - a = increase in encoding level
 - c = corrected or revised
 - d = deleted
 - n = new
 - p = increase in encoding from prepublication (previous CIP)
- 06 Type of record
 - a = language material
 - c = printed music
 - d = manuscript music
 - e = cartographic material
 - f = manuscript cartographic material
 - g = projected medium
 - i = nonmusical sound recording
 - j = musical sound recording
 - k = 2-dimensional nonprojectable graphic
 - m = computer file
 - o = kit
 - p = mixed materials
 - r = 3-dimensional artifact or naturally occurring object
 - t = manuscript language material
- 07 Bibliographic level
 - a = monographic component part
 - b = serial component part
 - c = collection
 - d = subunit
 - i = integrating resource
 - m = monograph/item
 - s = serial
- 08 Type of control
 - # = no specified type
 - a = archival
- 09 Character coding scheme
 - # = MARC-8
 - a = UCS/Unicode
- 10 Indicator count (always "2")
- 11 Subfield code count (always "2")
- 12-16 Base address of data (calculated by the computer for each record)
- 17 Encoding level
 - # = full level

- 1 = full level, material not examined
- 2 = less-than-full level, material not examined
- 3 = abbreviated level
- 4 = core level
- 5 = partial (preliminary) level
- 7 = minimal level
- 8 = prepublication level (CIP)
- u = unknown
- z = not applicable
- 18 **Descriptive cataloging form**
 - # = non-ISBD
 - a = AACR2
 - i = ISBD
 - u = unknown
- 19 **Multipart resource record level**
 - # = Not specified or not applicable
 - a = Set
 - b = Part with independent title
 - c = Part with dependent title
- 20 **Length of the length-of-field portion** (always "4")
- 21 **Length of the starting-character-position portion** (always "5")
- 22 **Length of the implementation-defined portion** (always "0")
- 23 **Undefined** (always "0")

[\[Back to Top of Page \]](#)

Part X:

Field 008 for Books

Field 008 is used for Fixed Length Data Elements ("Fixed Field Codes"). There are 40 character positions in field 008, numbered from 00-39. Undefined positions must contain either a blank (#) or a fill character (|). Positions 00-17 and 35-39 are defined the same way for all media.

The information shown here for positions 18-34 applies only to books. For explanation of all the positions below and for positions 18-34 for other media, see the *MARC 21 Format for Bibliographic Data*.

Note that field 008 has no indicators or subfield codes.

- 00-05 **Date entered on file (YYMMDD)**,
where Y=year, M=month, and D=day
- 06 **Type of date/publication status:**
 - b = no dates given; B.C. date involved
 - e = detailed date
 - s = single known date/probable date
 - m = multiple dates
 - r = reprint/reissue date (Date 1) and original date (Date 2)

- n = dates unknown
- q = questionable date
- t = publication date and copyright date
- | = no attempt to code

07-10 Date 1/beginning date of publication

11-14 Date 2/ending date of publication

Date fields contain the year(s) of publication. The type of date(s) in these elements are specified in fixed field element 06: Type of date/publication status. (For further details, see the field 008 description in the *MARC 21 Format for Bibliographic Data*.)

15-17 Place of publication, production, or execution

For example:

pk# = Pakistan

cau = California (US)

(For a full list of codes used in these positions, see the *MARC Code List for Countries*.)

18-21 Illustrations (up to 4 codes):

- # = no illustrations
- a = illustrations
- b = maps
- c = portraits
- d = charts
- e = plans
- f = plates
- g = music
- h = facsimiles
- i = coats of arms
- j = genealogical tables
- k = forms
- l = samples
- m = phonodisc, phonowire, etc.
- o = photographs
- p = illuminations
- | = no attempt to code

22 Target audience:

- # = unknown or not specified
- a = preschool
- b = primary
- c = pre-adolescent
- d = adolescent
- e = adult
- f = specialized
- g = general
- j = juvenile
- | = no attempt to code

23 Form of item:

- # = none of the following
- a = microfilm

b = microfiche
c = microopaque
d = large print
f = braille
r = regular print reproduction
s = electronic
| = no attempt to code

24-27 Nature of contents (up to 4):

= no specified nature of contents
a = abstracts/summaries
b = bibliographies (is one or contains one)
c = catalogs
d = dictionaries
e = encyclopedias
f = handbooks
g = legal articles
i = indexes
j = patent document
k = discographies
l = legislation
m = theses
n = surveys of literature
o = reviews
p = programmed texts
q = filmographies
r = directories
s = statistics
t = technical reports
u = standards/specifications
v = legal cases and notes
w = law reports and digests
z = treaties
| = no attempt to code

28 Government publication:

= not a government publication
i = international intergovernmental
f = federal/national
a = autonomous or semi-autonomous component
s = state, provincial, territorial, dependent, etc.
m = multistate
c = multilocal
l = local
z = other type of government publication
o = government publication -- level undetermined
u = unknown if item is government publication
| = no attempt to code

29 Conference publication:

0 = not a conference publication
1 = conference publication

- 30 | = no attempt to code
 Festschrift:
 0 = not a festschrift
 1 = festschrift
 | = no attempt to code
- 31 **Index:**
 0 = no index
 1 = index present
 | = no attempt to code
- 32 **Undefined (since 1990)** (Earlier records may contain the values 0 or 1)
 # = Undefined
 | = no attempt to code
- 33 **Literary form:**
 0 = not fiction (not further specified)
 1 = fiction (not further specified)
 c = comic strips
 d = dramas
 e = essays
 f = novels
 h = humor, satires, etc.
 i = letters
 j = short stories
 m = mixed forms
 p = poetry
 s = speeches
 u = unknown
 | = no attempt to code
- 34 **Biography:**
 # = no biographical material
 a = autobiography
 b = individual biography
 c = collective biography
 d = contains biographical information
 | = no attempt to code
- 35-37 **Language:**
 A three-letter code. For example: eng fre ger spa rus ita

(For a full list of codes used in these positions, see the [MARC Code List for Languages](#).)

- 38 **Modified record:**
 # = not modified
 x = missing characters (because of characters unavailable in MARC character set)
 s = shortened
 d = "dashed-on" information omitted
 r = completely romanized/printed cards in script
 o = completely romanized/printed cards romanized
 | = no attempt to code
- 39 **Cataloging source:**

= national bibliographic agency
c = cooperative cataloging program
d = other sources
u = unknown
| = no attempt to code

[[Back to Top of Page](#)]

[[Back to Table of Contents](#)] -- [[Continue to Part 11](#)]



Library of Congress

[Library of Congress Help Desk](#) (10/27/2009)

APPENDIX C

Appendix C. Materials Considered

Exhibit	Description
0001	Excerpts of <i>The Windows Interface Guidelines for Software Design</i> (1995). Microsoft Press. (“ <i>Windows Guidelines Book</i> ”)
0001A	Excerpts of <i>Windows Guidelines Book</i> obtained from the Library of Congress.
0001B	Bibliographic and MARC records for <i>Windows Guidelines Book</i> , available at https://lccn.loc.gov/95000330 from the online catalog of the Library of Congress.
0001C	LOCIS_ Library of Congress Information System, archived by Internet Archive on May 24, 1997, available at https://web.archive.org/web/19970524160715/http://library.usask.ca/hyteln/et/us3/us373.html .
0001D	Profile of the Library of Congress, available at https://librarytechnology.org/library/2263 from “libraries.org,” a directory of libraries throughout the world.
0001E	Copyright registration record for <i>Windows Guidelines Book</i> , available at https://publicrecords.copyright.gov/detailed-record/voyager_15160964 from the Public Records System of the US Copyright Office.
0001F	Bibliographic and MARC records for <i>Windows Guidelines Book</i> , available at https://search.lib.umich.edu/catalog/record/990030260390106381?query=isbn:32590068 from the online catalog of the University of Michigan Libraries.
0001G	Profile of the University of Michigan Libraries, available at https://librarytechnology.org/library/2089 from “libraries.org,” a directory of libraries throughout the world.
0001H	Publisher webpage for <i>Windows Guidelines Book</i> , archived by the Internet Archive on April 22, 2000, available at

	https://web.archive.org/web/20000422051029/http://www1.fatbrain.com/a-sp/bookinfo/bookinfo.asp?theisbn=1556156790 .
0001I	Citations to <i>Windows Guidelines Book</i> , obtained from the ACM Digital Library.
0002	Gomberg, M., Evard, R., & Stacey, C. (1998, August). A comparison of large-scale software installation methods on NT and UNIX, in <i>Proceedings of the Large Installation System Administration of Windows NT Conference, August 5-8, 1998, Seattle, Washington, USA</i> , available at https://www.usenix.org/legacy/publications/library/proceedings/lisa-nt98/full_papers/gomberg/gomberg.pdf . (“Gomberg”)
0002A	Gomberg, M., Evard, R., & Stacey, C. (1998, August). A comparison of large-scale software installation methods on NT and UNIX, in <i>Proceedings of the Large Installation System Administration of Windows NT Conference, August 5-8, 1998, Seattle, Washington, USA</i> , pp. 37 – 48, obtained from the Pennsylvania State University Libraries.
0002B	Bibliographic and MARC records for <i>Proceedings of the Large Installation System Administration of Windows NT Conference</i> that contains <i>Gomberg</i> , available at https://catalog.libraries.psu.edu/catalog/1712083 from the online catalog of the Pennsylvania State University Libraries.
0002C	Penn State Libraries Information Access Systems, archived by Internet Archive on February 9, 1998, available at https://web.archive.org/web/19980209162244/http://www.lias.psu.edu/ .
0002D	Bibliographic and MARC records for <i>Proceedings of the Large Installation System Administration of Windows NT Conference</i> that contains <i>Gomberg</i> , available at https://primo.lib.umn.edu/permalink/01UMN_INST/1msd8mg/alma9942259960001701 from the online catalog of the University of Minnesota Libraries.

0002E	Profile of the University of Minnesota Libraries, available at https://librarytechnology.org/library/1024 from “libraries.org,” a directory of libraries throughout the world.
0002F	USENIX Conference Proceedings website, archived on January 16, 1999, available at https://web.archive.org/web/19990116232824/http://www.usenix.org/publications/library/proceedings/index.html .
0002G	Technical Program of LISA-NT 1998, archived on August 23, 1999, available at https://web.archive.org/web/19990823080333/http://www.usenix.org/publications/library/proceedings/lisa-nt98/technical.html .
0002H	USENIX webpage for <i>Gomberg</i> abstract, archived on November 13, 1999, at https://web.archive.org/web/19991113081156/http://www.usenix.org/publications/library/proceedings/lisa-nt98/gomberg.html .
0002I	Citations to <i>Gomberg</i> , obtained from Google Scholar and Semantic Scholar.
App. A	Hsieh-Yee CV
App. B	Part VII to X of <i>Understanding MARC Bibliographic: Machine-Readable Cataloging</i> , available at https://www.loc.gov/marc/umb/um07to10.html from the Library of Congress.

Exhibit 0001

Microsoft® Professional Reference

Covers
Windows 95 &
Windows NT™!



The Windows® Interface Guidelines for Software Design



Start

Microsoft Press

THE WINDOWS® INTERFACE GUIDELINES FOR SOFTWARE DESIGN

Here are the Microsoft® guidelines for creating well-designed, visually and functionally consistent user interfaces for applications that run on the Microsoft Windows operating system. **THE WINDOWS INTERFACE GUIDELINES FOR SOFTWARE DESIGN** is an essential handbook for all programmers and designers working with the latest release of Windows, regardless of experience level or development tools used.

This book covers the basic principles of user interface design and design methodologies and specifies how data-centered concepts such as objects and properties can be applied to interface design. It includes detailed information on mouse, keyboard, and pen interaction and on how to use the common interface elements supplied by the system. This book also includes information about special considerations such as network computing, international users, and users with disabilities.

Topics include:

- *General input techniques*—navigation, selection, viewing, editing, and creation, including both command and direct manipulation methods such as drag and drop.
- *Windows*—primary and secondary types and their components, including property sheets, dialog boxes, message boxes, palette windows, and pop-up windows.
- *Menus, controls, and toolbars*—types and their components and when to use them.
- *Microsoft OLE*—how to support design interfaces for OLE embedded and linked objects, visual editing, and other forms of activation.
- *User assistance*—how to use contextual forms of help, including tooltips and wizards.
- *Integration with the system*—designing your software so that its interface functions and operates consistently with Windows.
- *Visual design*—effective use of color, layout, fonts, and graphics.

If you are developing or are considering developing applications for Microsoft Windows,

THE WINDOWS INTERFACE GUIDELINES FOR SOFTWARE DESIGN is a key resource.

U.S.A. \$29.95

U.K. £27.49

Canada \$39.95

[Recommended]

Microsoft Press

Operating Systems/Windows/Programming
ISBN 1-55615-679-0



X004GXB7ZP

New - New Windows
Interface

9 781556 156795

PUBLISHED BY
Microsoft Press
A Division of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Copyright © 1995 by Microsoft Corporation

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Library of Congress Cataloging-in-Publication Data
The Windows interface guidelines for software design.

p. cm.
Includes index.

ISBN 1-55615-679-0

1. Microsoft Windows (Computer file) 2. User interfaces (Computer systems) 3. Computer software--Development. I. Microsoft Corporation.

QA76.76.W56W553 1995
005.265--dc20

95-330
CIP

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 QE QE 0 9 8 7 6 5

Distributed to the book trade in Canada by Macmillan of Canada, a division of Canada Publishing Corporation.

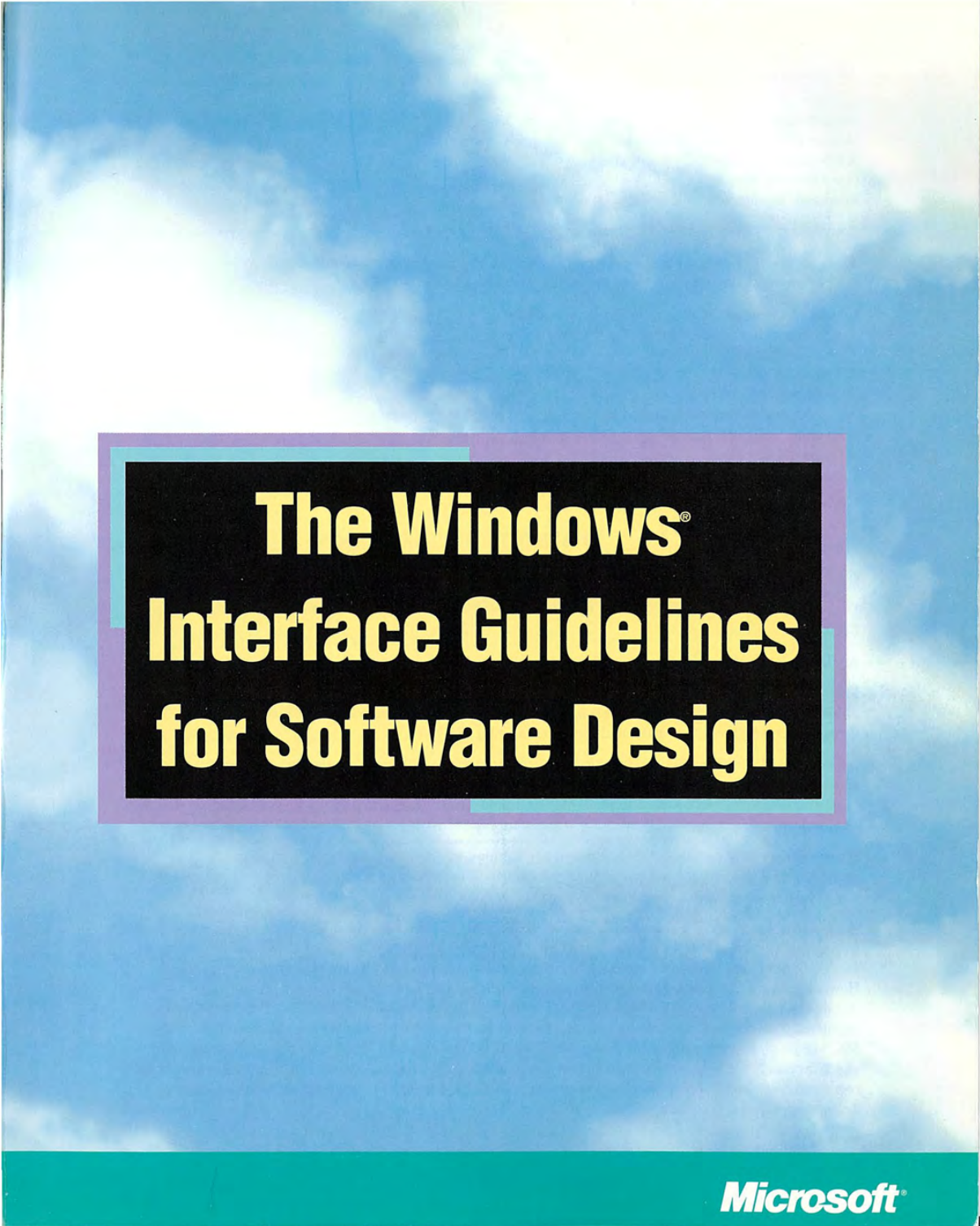
A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office. Or contact Microsoft Press International directly at fax (206) 936-7329.

Information in this document is subject to change without notice and does not represent a commitment on the part of Microsoft Corporation. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property rights.

Adobe, PostScript, and TIFF are trademarks of Adobe Systems, Inc. Apple and TrueType are registered trademarks of Apple Computer, Inc. Borland and Quattro are registered trademarks of Borland International, Inc. Frutiger is a registered trademark of Eltra Corporation. HP and LaserJet are registered trademarks of Hewlett-Packard Company. Backup was developed for Microsoft by Colorado Memory Systems, Inc., a division of Hewlett-Packard Company. HyperTerminal is a trademark of Hilgraeve, Inc. 1-2-3 and Lotus are registered trademarks of Lotus Development Corporation. Microsoft, Microsoft Press, Microsoft Press logo, MS, MS-DOS, PowerPoint, Visual Basic, Windows, Windows logo, and XENIX are registered trademarks and Windows NT is a trademark of Microsoft Corporation. Arial, Bodoni, Swing, and Times New Roman are registered trademarks of The Monotype Corporation PLC. Paintbrush is a trademark of Wordstar Atlanta Technology Center.



The Windows[®] Interface Guidelines for Software Design

Microsoft[®]

Contents

Introduction

What's New	xv
How to Use This Guide	xvi
How to Apply the Guidelines	xvii
Conventions Used in This Guide	xviii

PART I FUNDAMENTALS OF DESIGNING USER INTERACTION

Chapter 1 Design Principles and Methodology

User-Centered Design Principles	3
User in Control	3
Directness	4
Consistency	5
Forgiveness	6
Feedback	7
Aesthetics	7
Simplicity	8
Design Methodology	9
A Balanced Design Team	9
The Design Cycle	9
Usability Assessment in the Design Process	12
Understanding Users	15
Design Tradeoffs	16

Contents

Chapter 2 Basic Concepts

Data-Centered Design	17
Objects as Metaphor	18
Object Characteristics	18
Relationships	19
Composition	19
Persistence	20
Putting Theory into Practice	20

Chapter 3 The Windows Environment

The Desktop	23
The Taskbar	24
The Start Button	25
Window Buttons	26
The Status Area	26
Icons	26
Windows	28

Chapter 4 Input Basics

Mouse Input	29
Mouse Pointers	29
Mouse Actions	31
Keyboard Input	32
Text Keys	33
Access Keys	33
Mode Keys	34
Shortcut Keys	35
Pen Input	37
Pen Pointers	39
Pen Gestures	40
Pen Recognition	41
Ink Input	41
Targeting	42

Chapter 5 General Interaction Techniques

Navigation	43
Mouse and Pen Navigation	44
Keyboard Navigation	44
Selection	45
Selection Feedback	46
Scope of Selection	47
Hierarchical Selection	47
Mouse Selection	48
Pen Selection	55
Keyboard Selection	56
Selection Shortcuts	57
Common Conventions for Supporting Operations	58
Operations for a Multiple Selection	58
Default Operations and Shortcut Techniques	59
View Operations	59
Editing Operations	62
Editing Text	62
Handles	63
Transactions	64
Properties	65
Pen-Specific Editing Techniques	66
Transfer Operations	72
Command Method	74
Direct Manipulation Method	77
Transfer Feedback	83
Specialized Transfer Commands	86
Shortcut Keys for Transfer Operations	87
Creation Operations	87
Copy Command	87
New Command	88
Insert Command	88
Using Controls	88
Using Templates	88
Operations on Linked Objects	89

PART II WINDOWS INTERFACE COMPONENTS

Chapter 6 Windows

Common Types of Windows	95
Primary Window Components	95
Window Frames	96
Title Bars	96
Title Bar Icons	97
Title Text	98
Title Bar Buttons	101
Basic Window Operations	103
Activating and Deactivating Windows	103
Opening and Closing Windows	104
Moving Windows	106
Resizing Windows	106
Scrolling Windows	109
Splitting Windows	116

Chapter 7 Menus, Controls, and Toolbars

Menus	121
The Menu Bar and Drop-down Menus	121
Common Drop-down Menus	124
Pop-up Menus	126
Pop-up Menu Interaction	128
Common Pop-up Menus	129
Cascading Menus	133
Menu Titles	134
Menu Items	135
Controls	140
Buttons	141
List Boxes	149
Text Fields	157
Other General Controls	163
Pen-Specific Controls	169
Toolbars and Status Bars	172
Interaction with Controls in Toolbars and Status Bars	173
Support for User Options	174
Toolbar and Status Bar Controls	175
Common Toolbar Buttons	176

Chapter 8 Secondary Windows

Characteristics of Secondary Windows	179
Appearance and Behavior	179
Window Placement	183
Modeless vs. Modal	183
Default Buttons	184
Navigation in Secondary Windows	185
Validation of Input	187
Property Sheets and Inspectors	187
Property Sheet Interface	188
Property Sheet Commands	190
Closing a Property Sheet	191
Property Inspectors	191
Properties of a Multiple Selection	192
Properties of a Heterogeneous Selection	193
Properties of Grouped Items	193
Dialog Boxes	193
Dialog Box Commands	194
Layout	194
Common Dialog Box Interfaces	195
Palette Windows	207
Message Boxes	209
Title Bar Text	209
Message Box Types	209
Command Buttons in Message Boxes	212
Message Box Text	213
Pop-up Windows	215

PART III DESIGN SPECIFICATIONS AND GUIDELINES

Chapter 9 Window Management

Single Document Window Interface	219
Multiple Document Interface	220
Opening and Closing MDI Windows	222
Moving and Sizing MDI Windows	223
Switching Between MDI Child Windows	225
MDI Alternatives	225
Workspaces	227
Workbooks	229
Projects	231
Selecting a Window Model	233
Presentation of Object or Task	233
Display Layout	234
Data-Centered Design	235
Combination of Alternatives	235

Chapter 10 Integrating with the System

The Registry	237
Registering Application State Information	238
Registering Application Path Information	241
Registering File Extensions	242
Supporting Creation	249
Registering Icons	250
Registering Commands	251
Enabling Printing	253
Registering OLE	253
Registering Shell Extensions	254
Supporting the Quick View Command	256
Registering Sound Events	257
Installation	257
Copying Files	257
Providing Access to Your Application	260
Designing Your Installation Program	260

Installing Fonts	262
Installing Your Application on a Network	262
Uninstalling Your Application	263
Supporting AutoPlay	264
System Naming Conventions	266
Taskbar Integration	268
Taskbar Window Buttons	268
Status Notification	269
Message Notification	270
Application Desktop Toolbars	271
Full-Screen Display	272
Recycle Bin Integration	273
Control Panel Integration	273
Adding Control Panel Objects	273
Adding to the Passwords Object	273
Plug and Play Support	275
System Settings and Notification	275
Modeless Interaction	276

Chapter 11 Working with OLE Embedded and OLE Linked Objects

The Interaction Model	277
Creating OLE Embedded and OLE Linked Objects	279
Transferring Objects	279
Inserting New Objects	285
Displaying Objects	290
Selecting Objects	293
Accessing Commands for Selected Objects	295
Activating Objects	297
Outside-in Activation	297
Inside-out Activation	297
Container Control of Activation	298
OLE Visual Editing of OLE Embedded Objects	300
The Active Hatched Border	304

Contents

Menu Integration	305
Keyboard Interface Integration	308
Toolbars, Frame Adornments, and Palette Windows	310
Opening OLE Embedded Objects	313
Editing an OLE Linked Object	316
Automatic and Manual Updating	318
Operations and Links	319
Types and Links	320
Link Management	320
Accessing Properties of OLE Objects	321
The Properties Command	321
The Links Command	324
Converting Types	326
Using Handles	329
Undo Operations for Active and Open Objects	330
Displaying Messages	332
Object Application Messages	332
OLE Linked Object Messages	334
Status Line Messages	336

Chapter 12 User Assistance

Contextual User Assistance	339
Context-Sensitive Help	339
Guidelines for Writing Context-Sensitive Help	342
Tooltips	343
Status Bar Messages	344
Guidelines for Writing Status Bar Messages	345
The Help Command Button	346
Task-Oriented Help	347
Task Topic Windows	347
Guidelines for Writing Task Help Topics	348
Shortcut Buttons	349

Reference Help	350
The Reference Help Window	351
Guidelines for Writing Reference Help	352
The Help Topics Browser	354
The Help Topics Tabs	354
Wizards	358
Guidelines for Designing Wizards	359
Guidelines for Writing Text for Wizard Pages	363

Chapter 13 Visual Design

Visual Communication	365
Composition and Organization	366
Color	368
Fonts	370
Dimensionality	371
Design of Visual Elements	372
Basic Border Styles	372
Window Border Style	373
Button Border Styles	374
Field Border Style	375
Status Field Border Style	376
Grouping Border Style	376
Visual States for Controls	377
Layout	384
Font and Size	384
Capitalization	387
Grouping and Spacing	388
Alignment	389
Placement	389
Design of Graphic Images	390
Icon Design	391
Pointer Design	394
Selection Appearance	395
Highlighting	396
Handles	397
Transfer Appearance	398
Open Appearance	399
Animation	400

Chapter 14 Special Design Considerations

Sound	401
Accessibility	403
Types of Disabilities	404
Types of Accessibility Aids	406
Compatibility with Screen Review Utilities	408
The User's Point of Focus	411
Timing and Navigational Interfaces	411
Color	412
Keyboard and Mouse Interface	413
Documentation, Packaging, and Support	414
Usability Testing	414
Internationalization	415
Text	416
Graphics	417
Keyboards	418
Character Sets	419
Formats	419
Layout	420
References to Unsupported Features	420
Network Computing	421
Leverage System Support	421
Client-Server Applications	421
Shared Data Files	422
Record Processing	422
Telephony	423
Microsoft Exchange	424
Coexisting with Other Information Services	424
Adding Menu Items and Toolbar Buttons	424
Supporting Connections	425
Installing Information Services	425

PART IV APPENDICES

Appendix A Mouse Interface Summary

Interaction Guidelines for Common	
Unmodified Mouse Actions	429
Interaction Guidelines for Using the SHIFT Key	
to Modify Mouse Actions	431
Interaction Guidelines for Using the CTRL Key	
to Modify Mouse Actions	435

Appendix B Keyboard Interface Summary

Common Navigation Keys	437
Common Shortcut Keys	438
Windows Keys	439
Accessibility Keys	440
Access Key Assignments	441

Appendix C Guidelines Summary

General Design	443
Design Process	444
Input and Interaction	444
Windows	445
Control Usage	446
Integration	447
User Assistance	448
Visual Design	448
Sound	449
Accessibility	449
International Users	450
Network Users	450

Contents

Appendix D Supporting Specific Versions of Microsoft Windows

Microsoft Windows 3.1	451
Microsoft Windows NT 3.51	453

Appendix E International Word Lists

International Word Lists	455
--------------------------------	-----

Glossary	511
----------------	-----

Bibliography	523
--------------------	-----

Index	527
-------------	-----

Introduction

Welcome to *The Windows Interface Guidelines for Software Design*, an indispensable guide to designing software that runs with the Microsoft® Windows® operating system. The design of your software's interface, more than anything else, affects how a user experiences your product. This guide promotes good interface design and visual and functional consistency within and across Windows-based applications.

What's New

Continuing the direction set by Microsoft OLE, the enhancements in the Windows user interface provide a design evolution from the basic and graphical to the more object oriented — that is, from an application-centered interface to a more data-centered one. In response, developers and designers may need to rethink the interface of their software — the basic components and the respective operations and properties that apply to them. This is important because, from a user's perspective, applications have become less the primary focus and more the engines behind the objects in the interface. Users can now interact with data without having to think about applications, allowing them to better concentrate on their tasks.

When adapting your existing Windows-based software, make certain you consider the following important design aspects:

- Title bar text and icons
- Property sheets
- Transfer model (including drag and drop)

Introduction

- Pop-up menus
- New controls
- Integration with the system
- Help interface
- OLE embedding and OLE linking
- Visual design of windows, controls, and icons
- Window management
- Presentation of minimized windows

These elements are covered in depth throughout this guide.

How to Use This Guide

This guide is intended for those who are designing and developing Windows-based software. It may also be appropriate for those interested in a better understanding of the Windows environment and the human-computer interface principles it supports. The content of the guide covers the following areas:

- Basic design principles and process — fundamental design philosophy, assumptions about human behavior, design methodology, and concepts embodied in the interface.
- Interface elements — descriptive information about the various components in the interface as well as when and how to use them.
- Design details — specific information about the details of effective design and style when using the elements of the interface.
- Additional information — summary and quick reference information, a bibliography, a comprehensive word list in numerous languages to assist in product localization, and a glossary.

This guide focuses on the design and elements of an application's user interface. Although an occasional technical reference is included, this guide does not generally cover detailed information about technical implementation or application programming interfaces (APIs), because there are many different types of development tools that you can use to develop software for Windows. The documentation included with the Microsoft® Win32® Software Development Kit (SDK) is one source of information about specific APIs.

How to Apply the Guidelines

This guide promotes visual and functional consistency within and across the Windows operating system. Although following these guidelines is encouraged, you are free to adopt the guidelines that best suit your software. However, by following these guidelines, you enable users to transfer their skills and experience from one task to the next and to learn new tasks easily. In addition, evolution toward data-centered design breaks down the lines between traditional application domains, making inconsistencies in the interface more obvious and distracting to users.

Conversely, adhering to the design guidelines does not guarantee usability. The guidelines are valuable tools, but they must be combined with other factors as part of an effective software design process, such as application of design principles, task analysis, prototyping, and usability evaluation.

You may extend these guidelines, provided that you do so in the spirit of the principles on which they are based, and maintain a reasonable level of consistency with the visual and behavioral aspects of the Windows interface. In general, avoid adding new elements or behaviors unless the interface does not otherwise support them. More importantly, avoid changing an existing behavior for common elements. A user builds up expectations about the workings of an interface. Inconsistencies not only confuse the user, they also add unnecessary complexity.

Introduction

These guidelines supersede those issued for Windows version 3.1 and all previous releases and are specific to the development of applications designed for Microsoft® Windows®, Microsoft® Windows NT™ Workstation, and Microsoft® Windows NT Server. There is no direct relationship between these guidelines and those provided for other operating systems.

For more information about special considerations concerning developing applications for both Windows 95 and Windows NT operating system, see Appendix D, “Supporting Specific Versions of Windows.”

Conventions Used in This Guide

The following conventions are used throughout this guide.

Convention	Indicates
	A reference to related topics in this guide or other books that provide more information about the topic.
	Additional or special information about the topic.
SMALL CAPITAL LETTERS	Names of keys on the keyboard — for example, SHIFT, CTRL, or ALT.
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another — for example, CTRL+P or ALT+F4.
<i>Italic text</i>	New terms and variable expressions, such as parameters.
Bold text	Win32 API keywords and registry key entries.
Registry text	Examples of registry entries.
[]	Optional information.



Integrating with the System

Users appreciate seamless integration between the system and their applications. This chapter covers information about integrating your software with the system and how to extend its features, including using the registry to store information about your application, installing your application, using appropriate naming conventions, and supporting shell features, such as the taskbar, Control Panel, and Recycle Bin.

This chapter is only intended to provide an overview. Details required for some conventions go beyond the scope of this guide. For information about these conventions, see the documentation included in the Microsoft Win32 Software Development Kit (SDK). In addition, some of these conventions and features may not be supported in all releases. For more information about specific releases, see Appendix D, “Supporting Specific Versions of Windows.”

The Registry

Windows provides a special repository called the *registry* that serves as a central configuration database for user-, application-, and computer-specific information. Although the registry is not intended for direct user access, the information placed in it affects your application’s user interface. Registered information determines the icons,

commands, and other features displayed for files. The registry also makes it easier to manage and support configuration information used by your application and eliminates redundant information stored in different locations.

The registry is a hierarchical structure. Each node in the tree is called a key. Each key can contain subkeys and data entries called values. Key names cannot include a space, backslash (\), or wildcard character (* or ?). In the **HKEY_CLASSES_ROOT** key, names beginning with a period (.) are reserved for special syntax (filename extensions), but you can include a period within a key name. The name of a subkey must be unique with respect to its parent key. Key names are not localized into other languages, although their values may be.

A key can have any number of values. A value entry has three parts: the name of the value, its data type, and the value itself. Value entries larger than 2048 bytes should be stored as files with their filenames stored in the registry.

When the user installs your application, register keys for where application data is stored, for filename extensions, icons, shell commands, OLE registration data, and for any special extensions. To register your application's information, you can create a registration file and use the Registry Editor to merge this file into the system registry. You can also use other utilities that support this function, or use the system-supplied registry functions to access or manipulate registry data.



The example registry entries in this chapter represent only the hierarchical relationship of the keys. For more information about the registry and registry file formats, see the documentation included in the Win32 SDK.



To use memory most efficiently, the system stores only the registry entries that have been installed and that are required for operation. Applications should never fail to write a registry entry because it is not already installed. To ensure this happens, use registry creation functions when adding an entry.

Registering Application State Information

Use the registry to store state information for your application. Typically, the data you store here will be information you may have stored in initialization (.INI) files in previous releases of Windows. Create subkeys under the **Software** subkey in the **HKEY_LOCAL_MACHINE** and **HKEY_CURRENT_USER** keys that include information about your application.

```

HKEY_LOCAL_MACHINE
  Software
    CompanyName
    ProductName
    Version
...
HKEY_CURRENT_USER
  Software
    CompanyName
    ProductName
    Version

```

Use your application's **HKEY_LOCAL_MACHINE** entry as the location to store computer-specific data and the **HKEY_CURRENT_USER** entry to store user-specific data. The latter key allows you to store settings to tailor your application for individual users working with the same computer. Under your application's subkey, you can define your own structure for the information. Although the system still supports initialization files for backward compatibility, use the registry wherever possible to store your application's state information instead.

Use these keys to save your application's state whenever appropriate, such as when the user closes its primary window. In most cases, it is best to restore a window to its previous state when the user reopens it.

When the user shuts down the system with your application's window open, you may optionally store information in the registry so that the application's state is restored when the user starts up Windows. (The system does this for folders.) To have your application's state restored, store your window and application state information under its registry entries when the system notifies your application that it is shutting down. Store the state information in your application's entries under **HKEY_CURRENT_USER** and add a value name–value pair to the **RunOnce** subkey that corresponds to your application. When the user restarts the system, it runs the command line you supply. Once your application runs, you can use the data you stored to restore its state.

HKEY_CURRENT_USER

Software

Microsoft

Windows

CurrentVersion

RunOnce *application identifier = command line*

If you have multiple instances open, you can include value name entries for each or consolidate them as a single entry and use command-line switches that are most appropriate for your application. For example, you can include entries like the following.

WordPad Document 1 = C:\Program Files\Wordpad.exe Letter to Bill /restore
WordPad Document 2 = C:\Program Files\Wordpad.exe Letter to Paul /restore
Paint = C:\Program Files\Paint.exe Abstract.bmp Cubist.bmp

As long as you provide a valid command-line string that your application can process, you can format the entry in a way that best fits your application.

You can also include a **RunOnce** entry under the **HKEY_LOCAL_MACHINE** key. When using this entry, however, the system runs the application before starting up. You can use this entry for applications that may need to query the user for information that affects how Windows starts. Just remember that any entry here will affect all users of the computer.

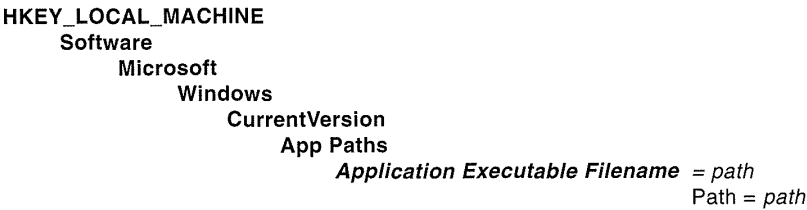
RunOnce entries are automatically removed from the registry once the system starts up. Therefore, you need not remove or update the entries, but your application must always save its state when the user shuts down the system. The system also supports a **Run** subkey in both the **HKEY_CURRENT_USER** and **HKEY_LOCAL_MACHINE** keys. The system runs any value name entries under this subkey after the system starts up, but does not remove those entries from the registry. For example, a virus check program can be installed to run automatically after the system starts up. You can also support this functionality by placing a file or shortcut to a file in the Startup folder. The registry stores the location of the Startup folder, as a value in **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders**.



The system's ability to restore an application's state depends on the availability of the application and its data files. If they have been deleted or the user has logged in over the network where the same files are not available, the system may not be able to restore the state.

Registering Application Path Information

The system supports “per application” paths. If you register a path, Windows sets the PATH environment variable to be the registered path when it starts your application. You set your application’s path in the **App Paths** subkey under the **HKEY_LOCAL_MACHINE** key. Create a new key using your application’s executable filename as its name. Set this key’s Default value to the path of your executable file. The system uses this entry to locate your application if it fails to find it in the current path; for example, if the user chooses the Run command on the Start menu and only includes the filename of the application, or if a shortcut icon doesn’t include a path setting. To identify the location of dynamic-link libraries placed in a separate directory, you can also include another value entry called Path and set its value to the path of your dynamic-link libraries.



The system will automatically update the path and default entries if the user moves or renames the application’s executable file using the system shell user interface.

Register any system-wide shared dynamic-link libraries in a subkey under a **SharedDLLs** subkey of **HKEY_LOCAL_MACHINE** key. If the file already exists, increment the entry’s usage count index. For more information about the usage count index, see the section, “Installation,” later in this chapter.



Registering File Extensions

If your application creates and maintains files, register entries for the file types that you expose directly to users and that you want users to be able to easily differentiate. For every file type you register, include at least two entries: a filename-extension key entry and an application (class) identification key entry.

If you do not register an extension for a file type, it will be displayed with the system's generic file object icon, as shown in Figure 10.1, and its extension will always be displayed. In addition, the user will not be able to double-click the file to open it. (Open With will be the icon's default command.)



Figure 10.1 System-generated icons for unregistered types

The Filename Extension Key

The filename extension entry maps a filename extension to an application identifier. To register an extension, create a subkey in the **HKEY_CLASSES_ROOT** key using the three-letter extension (including a period) and set its value to an application identifier.

HKEY_CLASSES_ROOT

.ext = *ApplicationIdentifier*

For the value of the application identifier (also known as programmatic identifier or Prog ID), use a string that uniquely identifies a given class. This string is used internally by the system and is not exposed directly to users (unless explicitly exported with a special registry utility); therefore, you need not localize this entry.

Avoid assigning multiple extensions to the same application identifier. To ensure that each file type can be distinguished by the user, define each extension such that each has a unique application identifier. If you have utility files that the user does not interact with directly, you should still register an extension (and icon) for them, preferably the same extension so that they can be identified. In addition, mark them with the hidden file attribute.

The system provides no arbitration for applications that use the same extensions. So define unique identifiers and check the registry to avoid writing over and replacing existing extension entries, a practice which may seriously affect the user's existing files. More specifically, avoid registering an extension that conflicts or redefines the common filename extensions used by the system. Examples of these extensions are shown in Table 10.1.

Table 10.1 Common Filename Extensions Supported by Windows

Extension	Type description
386	Windows virtual device driver
3GR	Screen grabber for MS-DOS-based applications
ACM	Audio compression manager driver
ADF	Administration configuration files
ANI	Animated pointer
AVI	Video clip
AWD	FAX viewer document
AWP	FAX key viewer
AWS	FAX signature viewer
BAK	Backed-up file
BAT	MS-DOS batch file
BFC	Briefcase
BIN	Binary data file
BMP	Picture (Windows bitmap)
CAB	Windows Setup file
CAL	Windows Calendar file
CDA	CD audio track
CFG	Configuration file

(Continued)

Extension	Type description
CNT	Help contents
COM	MS-DOS – based application
CPD	FAX cover page
CPE	FAX cover page
CPI	International code page
CPL	Control Panel extension
CRD	Windows Cardfile document
CSV	Command-separated data file
CUR	Cursor (pointer)
DAT	System data file
DCX	FAX viewer document
DLL	Application extension (dynamic-link library)
DOC	WordPad document
DOS	MS-DOS file (also extension for NDIS2 net card and protocol drivers)
DRV	Device driver
EXE	Application
FND	Saved search
FON	Font file
FOT	Shortcut to font
GR3	Windows 3.0 screen grabber
GRP	Program group file
HLP	Help file
HT	HyperTerminal™ file
ICM	ICM profile
ICO	Icon
IDF	MIDI instrument definition
INF	Setup information
INI	Initialization file (configuration settings)

(Continued)

Extension	Type description
KBD	Keyboard layout
LGO	Windows logo driver
LIB	Static-link library
LNK	Shortcut
LOG	Log file
MCI	MCI command set
MDB	File viewer extension
MID	MIDI sequence
MIF	MIDI instrument file
MMF	Microsoft Mail message file
MMM	Animation
MPD	Mini-port driver
MSG	Microsoft® Exchange mail document
MSN	Microsoft Network home base
NLS	Natural language services driver
PAB	Microsoft Exchange personal address book
PCX	Bitmap picture (PCX format)
PDR	Port driver
PF	ICM profile
PIF	Shortcut to MS-DOS–based application
PPD	PostScript® printer description file
PRT	Printer formatted file (result of Print to File option)
PST	Microsoft Exchange personal information store
PWL	Password list
QIC	Backup set for Microsoft Backup
REC	Windows Recorder file
REG	Application registration file
RLE	Picture (RLE format)
RMI	MIDI sequence

(Continued)

Extension	Type description
RTF	Document (rich-text format)
SCR	Screen saver
SET	File set for Microsoft Backup
SHB	Shortcut into a document
SHS	Scrap
SPD	PostScript printer description file
SWP	Virtual memory storage
SYS	System file
TIF	Picture (TIFF® format)
TMP	Temporary file
TRN	Translation file
TSP	Windows telephony service provider
TTF	TrueType® font
TXT	Text document
VBX	Microsoft Visual Basic® control file
VER	Version description file
VXD	Virtual device driver
WAV	Sound wave
WPC	WordPad file converter
WRI	Windows Write document

It is a good idea to investigate extensions commonly used by popular applications so you can avoid creating a new extension that might conflict with them, unless you intend to replace or superset the functionality of those applications.

The Application Identifier Key

The second registry entry you create for a file type is its class-definition (Prog ID) key. Using the same string as the application identifier you used for the extension's value, create a key, and assign a type name as the value of the key.

HKEY_CLASSES_ROOT

```
.ext = ApplicationIdentifier  
ApplicationIdentifier = Type Name
```

Under this key, you specify shell and OLE properties of the class. Provide this entry even if you do not have any extra information to place under this key; doing so provides a label for users to identify the file type. In addition, you use this entry to register the icon for the file type.

Define the type name (also known as the `MainUserTypeName`) as the human-readable form of its application identifier or class name. It should convey to the user the object's name, behavior, or capability. A type name can include all of the following elements:

1. *Company Name*
Communicates product identity.
2. *Application Name*
Indicates which application is responsible for activating a data object.
3. *Data Type*
Indicates the basic category of the object (for example, drawing, spreadsheet, or sound). Limit the number of characters to a maximum of 15.
4. *Version*
When there are multiple versions of the same basic type, for upgrading purposes, you may want to include a version number to distinguish types.

When defining your type name, use title capitalization. The name can include up to a maximum of 40 characters. Use one of the following three recommended forms:

1. *Company Name Application Name [Version] Data Type*
For example, Microsoft Excel Worksheet.
2. *Company Name-Application Name [Version] Data Type*
For cases when the company name and application are the same — for example, ExampleWare 2.0 Document.
3. *Company Name Application Name [Version]*
When the application sufficiently describes the data type — for example, Microsoft Graph.

These type names provide the user with a precise language for referring to objects. Because object type names appear throughout the interface, the user becomes conscious of an object's type and its associated behavior. However, because of their length, you may also want to include a short type name. A *short type name* is the data type portion of the full type name. Applications that support OLE always include a short type name entry in the registry. Use the short type name in drop-down and pop-up menus. For example, a Microsoft® Excel Worksheet is simply referred to as a “Worksheet” in menus.

To provide a short type name, add an **AuxUserType** subkey under the application's registered **CLSID** subkey (which is under the **CLSID** key).

```
HKEY_CLASSES_ROOT
    .ext = ApplicationIdentifier
...
    ApplicationIdentifier = Type Name
        CLSID = {CLSID identifier}
...
        CLSID
            {CLSID identifier}
                AuxUserType
                    2 = Short Type Name
```

If a short type name is not available for an object because the string was not registered, use the full type name instead. All controls that display the full type name must allocate enough space for 40 characters in width. By comparison, controls need only accommodate 15 characters when using the short type name.



For more information about registering type names and other information you should include under the **CLSID** key, see the OLE documentation included in the Win32 SDK.

Supporting Creation

The system supports the creation of new objects in system containers, such as folders and the desktop. Register information for each file type that you want the system to include. The registered type will appear on the New command that the system includes on menus for the desktop, folders, and the Open and Save As common dialog boxes. This provides a more data-centered design because the user can create a new object without having to locate and run the associated application.

To register a file type for inclusion, create a subkey using the Application Identifier under the extension's subkey in **HKEY_CLASSES_ROOT**. Under it, also create the **ShellNew** subkey.

HKEY_CLASSES_ROOT

.ext = ApplicationIdentifier

ApplicationIdentifier

ShellNew Value Name = Value

Assign a value entry to the **ShellNew** subkey with one of the four methods for creating a file with this extension.

Value name	Value	Result
NullFile	" "	Creates a new file of this type as a null (empty) file.
Data	<i>binary data</i>	Creates a new file containing the binary data.
FileName	<i>path</i>	Creates a new file by copying the specified file.
Command	<i>filename</i>	Carries out the command. Use this to run your own application code to create a new file (for example, run a wizard).

The system also will automatically provide a unique filename for the new file using the type name you register.

When using a Command value, place your application file (that creates the new file) in the directory that the system uses to store these files. To determine the path for that directory, check the setting for the Templates value in the **Shell Folders** subkey found in **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer**. Then you need only register the filename for the command.

Registering Icons

The system uses the registry to determine which icon to display for a specific file. You register an icon for every data file type that your application supports and that you want the user to be able to distinguish easily. Create a **DefaultIcon** subkey entry under the application identifier subkey you created and define its value as the filename containing the icon. Typically, you use the application's executable filename and the index of the icon within the file. The index value corresponds to the icon resource within the file. A positive number represents the icon's position in the file. A negative number corresponds to the inverse of the resource ID number of the icon. The icon for your application should always be the first icon resource in your executable file. The system always uses the first icon resource to represent executable files. This means the index value for your data files will be a number greater than 0.

HKEY_CLASSES_ROOT

ApplicationIdentifier = Type Name
DefaultIcon = path [,index]

Instead of registering the application's executable file, you can register the name of a dynamic link library file (.DLL), an icon file (.ICO), or bitmap file (.BMP) to supply your data file icons. If an icon does not exist or is not registered, the system supplies an icon derived from the icon of the file type's registered application. If no icon is available for the application, the system supplies a generic icon. These icons do not make your files uniquely identifiable, so design and register icons for both your application and its data file types. Include the following sizes: 16 x 16 pixel (16 color), 32 x 32 pixel (16 color), and 48 x 48 pixel (256 color).



For more information about designing icons, see Chapter 13, "Visual Design."

Registering Commands

Many of the commands found on icons, including Send To, Cut, Copy, Paste, Create Shortcut, Delete, Rename, and Properties, are provided by their container — that is, their containing folder or the desktop. But you must provide support for the icon's primary commands, also referred to as verbs, such as Open, Edit, Play, and Print. You can also register additional commands that apply to your file types, such as a What's This? command and even commands for other file types.

To add these commands, in the **HKEY_CLASSES_ROOT** key, you register a **shell** subkey and a subkey for each verb, and a **command** subkey for each menu command name.

HKEY_CLASSES_ROOT

```
ApplicationIdentifier = Type Name
  shell [ = default verb [, verb2 [...]]
    verb [ = Menu Command Name]
      command = pathname [parameters]
```

You can also register a DDE command string for a DDE command.

HKEY_CLASSES_ROOT

```
ApplicationIdentifier = Type Name
  shell [ = default verb [, verb2 [...]]
    verb [ = Menu Command Name]
      ddeexec = DDE command string
      Application = DDE Application Name
      Topic = DDE topic name
```

A verb is a language-independent name of the command. Applications may use it to invoke a specific command programmatically. The system defines Open, Print, Find, and Explore as standard verbs and automatically provides menu command names and appropriate access key assignments, localized in each international version of Windows. When you supply verbs other than these, provide menu command names localized for the specific version of Windows on which the application is installed. To assign a menu command name for a verb, make it the default value of the verb subkey.

The menu command names corresponding to the verbs for a file type are displayed to the user, either on a folder's File drop-down menu or pop-up menu for a file's icon. These appear at the top of the menu. You define the order of the menu commands by ordering the verbs in the value of the **shell** key. The first verb becomes the default command in the menu.

By default, capitalization follows how you enter format the menu command name value of the verb subkey. Although the system automatically capitalizes the standard commands (Open, Print, Explore, and Find), you can use the value of the menu command name to format the capitalization differently. Similarly, you use the menu command name value to set the access key for the menu command following normal menu conventions, prefixing the character in the name with an ampersand (&). Otherwise, the system sets the first letter of the command as the access key for that command.

To support user execution of a verb, provide the path for the application or a DDE command string. You can include command-line switches. For paths, include a %1 parameter. This parameter is an operational placeholder for whatever file the user selects.

For example, to register an Analyze command for an application that manages stock market information, the registry entries might look like the following.

```
HKEY_CLASSES_ROOT
    stockfile = Stock Data
        shell = analyze
            analyze = &Analyze
                command = C:\Program Files\Stock Analysis\Stock.exe /A
```

You may have different values for each command. You may assign one application to carry out the Open command and another to carry out the Print command, or use the same application for all commands.

Enabling Printing

If your file types are printable, include a Print verb entry in the **shell** subkey under **HKEY_CLASSES_ROOT**, following the conventions described in the previous section. This will display the Print command on the pop-up menu for the icon and on the File menu of the folder in which the icon resides when the user selects the icon. When the user chooses the Print command, the system uses the registry entry to determine what application to run to print the file.

Also register a Print To registry entry for the file types your application supports. This entry enables dragging and dropping of a file onto a printer icon. Although a Print To command is not displayed on any menu, the printer includes Print Here as the default command on the pop-up menu displayed when the user drag and drops a file on the printer using button 2.

In both cases, print the file, preferably, without opening the application's primary window. One way to do this is to provide a command-line switch that runs the application for handling the printing operation only (for example, WordPad.exe /p). In addition, display some form of user feedback that indicates whether a printing process has been initiated and, if so, its progress. For example, this feedback could be a modeless message box that displays, "Printing page *m* of *n* on *printer name*" and a Cancel button. You may also include a progress indicator control.

Registering OLE

Applications that support OLE use the registry as the primary means of defining class types, operations, and properties for data types supported by those applications. You store OLE registration information in the **HKEY_CLASSES_ROOT** key in subkeys under the **CLSID** subkey and in the class description's (Prog ID) subkey.



For more information about the specific registration entries for OLE, see the OLE documentation included in the Win32 SDK.

Registering Shell Extensions

Your application can extend the functionality of the operational environment provided by the system, also known as the shell, in a number of ways. A shell extension enhances the system by providing additional ways to manipulate file objects, by simplifying the task of browsing through the file system, or by giving the user easier access to tools that manipulate objects in the file system.

Every shell extension requires a *handler*, special application code (32-bit OLE InProc server implemented as a dynamic-link library) that implements subordinate functions. The types of handlers you can provide include:

- Pop-up (context) menu handlers: these add menu items to the pop-up menu for a particular file type.
- Drag handlers: these allow you to support the OLE data transfer conventions for drag and drop operations of a specific file type.
- Drop handlers: these allow you to carry out some action when the user drops objects on a specific type of file.
- Nondefault drag and drop handlers: these are pop-up menu handlers that the system calls when the user drags and drops an object by using mouse button 2.
- Icon handlers: these can be used to add per-instance icons for file objects or to supply icons for all files of a specific type.
- Property sheet handlers: these add pages to a property sheet that the shell displays for a file object. The pages can be specific to a class of files or to a particular file object.
- Copy-hook handlers: these are called when a folder or printer object is about to be moved, copied, deleted, or renamed by the user. The handler can be used to allow or prevent the operation.



Support for shell extensions may depend on the version of Windows installed. For more information about specific releases, see Appendix D, "Supporting Specific Versions of Windows."

You register the handler for a shell extension in the **HKEY_CLASSES_ROOT** key. The **CLSID** subkey contains a list of class identifier key values such as {00030000-0000-0000-C000-000000000046}. Each class identifier must also be a globally unique identifier.



For more information about creating handlers and class identifiers, see the OLE documentation included in the Win32 SDK.

You must also create a **shellex** subkey under the application's class identification entry in the **HKEY_CLASSES_ROOT** key.

HKEY_CLASSES_ROOT

ApplicationIdentifier = Type Name

Shell [= default verb [, verb2 [...]]

...

shellex

HandlerType

{*CLSID identifier*} = Handler Name

...

HandlerType = {*CLSID identifier*}

The shell also uses several other special keys, such as *****, **Folder**, **Drives**, and **Printers**, under **HKEY_CLASSES_ROOT**. You can use these keys to register extensions for system-supplied objects. For example, you may use the ***** key to register handlers that the shell calls whenever it creates a pop-up menu or property sheet for a file object, as in the following example.

HKEY_CLASSES_ROOT

*** = ***

shellex

ContextMenuHandlers

{00000000-1111-2222-3333-000000000001}

PropertySheetHandlers = SummaryInfo

{00000000-1111-2222-3333-000000000002}

IconHandler = {00000000-1111-2222-3333-000000000003}

The shell would use these handlers to add to the pop-up menus and property sheets of every file object. (The entries are intended only as examples, not literal entries.)

A pop-up menu handler may add commands to the pop-up menu of a file type, but it may not delete or modify existing menu commands. You can register multiple pop-up menu handlers for a file type. The order of the subkey entries determines the order of the items in the context menu. Handler-supplied menu items always follow registered command names.

Keep in mind that if you want to include a command on the pop-up menu of every file of a particular type, you do not need to create and register a pop-up menu handler. You can just use the normal means of registering commands for that type. Create a pop-up menu handler only when you want to provide a command only under specific conditions, such as the length of the file or its timestamp.

When registering an icon handler for providing per-instance icons for a file type, set the value for the **DefaultIcon** key to %1. This denotes that each file instance of this type can have a different icon.

Supporting the Quick View Command

The system includes support for fast, read-only views of many file types when the user chooses the Quick View command from the file object's menu. This allows the user to view files without opening the application.

If your file type is not supported, you can install a file parser that translates your file type into a format the system file viewer can read. Although this approach allows you to easily support viewers for your data file types, it limits the interaction options for your file types to those provided by the system. Alternatively, you can create your own file viewer, using the system-supplied interfaces. You can also register a file viewer for a file type already registered.

You can also support the Quick View command for data objects stored within your application's interface, either by supplying a specific viewer for your data types or by writing the data to a temporary file and then executing a file viewer and passing the temporary file as a parameter.



For more information about supporting the Quick View command and creating file viewers, see the documentation included in the Win32 SDK.

Registering Sound Events

Your application can register specific events to which the user can assign sound files so that when those events are triggered, the assigned sound file is played. To register a sound event, create a key under the **HKEY_CURRENT_USER** key.

HKEY_CURRENT_USER

AppEvents

Event Labels

EventName = *Event Name*

Set the value for *EventName* to a human-readable name.

Registering a sound event only makes it available in Control Panel so the user can assign a sound file. Your application must provide the code to process that event.

Installation

The following sections provide guidelines for installing your application's files. Applying these guidelines will help you reduce the clutter of irrelevant files when the user browses for a file. In addition, you'll reduce the redundancy of common files and make it easier for the user to update applications or the system software.

Copying Files

When the user installs your software, avoid copying files into the Windows directory (folder) or its System subdirectory. Doing so clutters the directory and may degrade system performance. Instead, create a single directory, preferably using the application's name, in the Program Files directory (or the location that the user chooses). In this directory, place the executable file. For example, if a program is named My Application, create a My Application subdirectory and place My Application.exe in that directory.

To locate the Program Files directory, check the `ProgramFilesDir` value in the **CurrentVersion** subkey under **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows**. The actual directory may not literally be named Program Files. For example, in international versions of Microsoft Windows, the directory name is appropriately localized. For networks that do not support the Windows long filename conventions, MS-DOS names may be used instead.

In your application's directory, create a subdirectory named **System** and place all support files that the user does not directly access in it, such as dynamic-link libraries and Help files. For example, place a support file called `My Application.dll` in the subdirectory `Program Files\My Application\System`. Hide the support files and your application's **System** directory and register its location using a **Path** value in the **App Paths** subkey under **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion**. Although you may place support files in the same directory as your application, placing them in a subdirectory helps avoid confusing the user and makes files easier to manage.

Applications can share common support files to reduce the amount of disk space consumed by duplication. If some non-user-accessed files of your application are shared as systemwide components (such as Visual Basic's `Vbrun300.dll`), place them in the **System** subdirectory of the directory where the user installs Windows. The process for installing shared files includes these logical steps:

1. Before copying the file, verify that it is not already present.
2. If the file is already present, compare its date and size to determine whether it is the same version as the one you are installing. If it is, increment the usage count in its corresponding registry entry.
3. If the file you are installing is not more recent, do not overwrite the existing version.
4. If the file is not present, copy it to the directory.

If you store a new file in the **System** directory installed by Windows, register a corresponding entry in the **SharedDLL** subkey under the **HKEY_LOCAL_MACHINE** key.



The system provides support services in `Ver.dll` for assisting you to do version verification. For more information about this utility, see the documentation included in the Win32 SDK.

If a file is shared, but only among your applications, create a subdirectory using your application's name in the Common Files subdirectory of the Program Files subdirectory and place the file there. To locate the Common Files directory, check the Common-FilesDir value in the **CurrentVersion** subkey of **HKEY_LOCAL_MACHINE\Software\Microsoft\Windows**. Alternatively, for “suite” style when multiple applications are bundled together, you can create a suite subdirectory in Program Files, where you place your executable files, and within that a System subdirectory with the support files shared only within the suite. In either case, register the path using the **Path** subkey under the **App Paths** subkey

When installing an updated version of the shared file, ensure that it is upwardly compatible before replacing the existing file. Alternatively, you can create a separate entry with a different filename (for example, Vbrun301.dll).

Name your executable file, dynamic-link libraries, and any other files that the user does not directly use, but that may be shared on a network, using conventional MS-DOS (8.3) names rather than long filenames. This will provide better support for users operating in environments where these files may need to be installed on network services that do not support the Windows long filename conventions.

Windows no longer requires Autoexec.bat and Config.sys files. Ensure that your application also does not require these files. Consider converting any MS-DOS device drivers to Windows virtual device drivers. The system supports dynamic loading of this type of device drivers, unlike MS-DOS device drivers which need to be loaded through Config.sys when starting the system. Similarly, because the registry allows you to register your application paths, your application does not require path information in Autoexec.bat.

In addition, do not make entries in Win.ini. Storing information in this file can make it difficult for the user to update or move your application. Also, avoid maintaining your application's own initialization file. Instead, use the registry. The registry provides conventions for storing most application and user settings. The registry provides greater flexibility allowing you to store information on a per machine or per user basis. It also supports accessing this information across a network.

Make certain you register the types supported by your application and the icons for these types along with your application's icons. In addition, register other application information, such as information required to enable printing.

Providing Access to Your Application

To provide easy user access to your application, place a shortcut icon to the application in the Programs folder. You can determine the path for this folder in **Shell Folders** subkey under **HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer**. This adds the entry to the submenu of the Programs menu of the Start button. Avoid adding entries for every application you might include in your software; this quickly overloads the menu. Optionally, you can allow the user to choose which icons to place in the menu. Avoid using a folder as your entry in the Programs menu, because this creates a multilevel hierarchy. Including a single entry makes it easier and simpler for a user to access your application.

Also consider the layout of files you provide with your application. Folders in Windows 95 and later releases provide much greater flexibility for file organization than did the Windows Program Manager. In addition to the recommended structure for your main executable file and its support files, you may want to create special folders for documents, templates, conversion tools, or other files that the user accesses directly.



You can create a "program group" entry in the Programs folder using the Windows 3.1 dynamic data exchange application programming interface (API). However, it is not recommended for applications installed with Microsoft Windows 95 and later releases, configured with the new shell user interface.

Designing Your Installation Program

Your installation program should offer the user different installation options such as:

- **Typical Setup:** installation that proceeds with the common defaults set, copying only the most common files. Make this the default setup option.
- **Compact Setup:** installation of the minimum files necessary to operate your application. This option is best for situations where disk space must be conserved — for example, on laptop computers. You can optionally add a Portable setup option for additional functionality designed especially for configurations on laptops, portables, and portables used with docking stations.

- **Custom Setup:** installation for the experienced user. This option allows the user to choose where to copy files and which options or features to include. This can include options or components not available for compact or typical setup.
- **CD-ROM Setup:** installation from a CD-ROM. This option allows users to select what files to install from the CD and allows them to run the remaining files directly from the CD.
- **Silent Setup:** installation using a command-line switch. This allows your setup program to run with a batch file

In addition to these setup options, your installation program should be a well-designed, Windows-based application and follow the conventions detailed in this guide and in the following guidelines:

- Supply a common response to every option so that the user can step through the installation process by confirming the default settings (that is, by pressing the ENTER key).
- Tell users how much disk space they will need before proceeding with installation. In the custom setup option, adjust the figure as the user chooses to include or exclude certain options. If there is not sufficient disk space, let the user know, but also give the user the option to override.
- Offer the user the option to quit the installation before it is finished. Keep a log of the files copied and the settings made so the canceled installation can be cleaned up easily.
- Ask the user to insert a disk only once during the installation. Lay out your files on disk so that the user does not have to reinsert the same disk multiple times.
- Provide a visual prompt and an audio cue when the user needs to insert the next disk.
- Support installation from any location. Do not assume that installation must be done from a logical MS-DOS drive (such as drive A). Design your installation program to support any valid universal naming convention (UNC) path.
- Provide a progress indicator message box to inform the user how far they are through the installation process.

If you are creating your own installation program, consider using the wizard control. Using this control and following the guidelines for wizards will result in a consistent interface for users.



For more information about designing wizards, see Chapter 12, “User Assistance.”

Naming your installation program Setup.exe or Install.exe (or localized equivalent) will allow the system to recognize the file. Place the file in the root directory of the disk the user inserts. This allows the system to automatically run your installation program when the user chooses the Install button in the Add/Remove Programs utility in Control Panel.

Installing Fonts

When installing fonts with your application on a local system, determine whether the font is already present. If it is, rename your font file — for example, by appending a number to the end of its filename. After copying a font file, register the font in the **Fonts** subkey.

Installing Your Application on a Network

If you create a client-server application so that multiple users access it from a network server, create separate installation programs: an installation program that allows the network administrator to prepare the server component of the application, and a client installation program that installs the client component files and sets up the settings to connect to the server. Design your client software so that an administrator can deploy it over the network and have it automatically configure itself when the user starts it.



For more information about designing client-server applications, see Chapter 14, “Special Design Considerations.” Additional information can also be found in the documentation included in Win32 SDK.

Because Windows may itself be configured to be shared on a server, do not assume that your installation program can store information in the main Windows directory on the server. In addition, shared application files should not be stored in the “home” directory provided for the user.

Design your installation program to support UNC paths. Also, use UNC paths for any shortcut icons you install in the Start Menu folder.

Uninstalling Your Application

The user may need to remove your application to recover disk space or to move the application to another location. To facilitate this, provide an uninstall program with your application that removes its files and settings. Remember to remove registry entries and shortcuts your application may have placed in the Start menu hierarchy. However, be careful when removing your application’s directory structure not to delete any user files (unless you confirm their removal with the user).

Your uninstall program should follow the conventions detailed in this guide and in the following guidelines:

- Display a window that provides the user with information about the progress of the uninstall process. You can also provide an option to allow the program to uninstall “silently” — that is, without displaying any information so that it can be used in batch files.
- Display clear and helpful messages for any errors your uninstall program encounters during the uninstall process.
- When uninstalling an application, decrement the usage count in the registry for any shared component — for example, a dynamic-link library. If the result is zero, give the user the option to delete the shared component with the warning that other applications may use this file and will not work if it is missing.

Registering your uninstall program will display your application in the list of the Uninstall page of the Add/Remove Program utility included with Windows. To register your uninstall program, add entries for your application to the **Uninstall** subkey.

```
HKEY_LOCAL_MACHINE
  Software
    Microsoft
      Windows
        CurrentVersion
          Uninstall
            ApplicationName DisplayName = Application Name
                               UninstallString = path [ switches ]
```

Both the **DisplayName** and **UninstallString** values must be supplied and be complete for your uninstall program to appear in the Add/Remove Program utility. The path you supply to **UninstallString** must be the complete command line used to carry out your uninstall program. The command line you supply should carry out the uninstall program directly rather than from a batch file or subprocess.

Supporting AutoPlay

Windows supports the ability to automatically run a file when the user inserts removable media that support insertion notification, such as CD-ROM, PCMCIA hard disks, or flash ROM cards. To support this feature, include a file named `Autorun.inf` in the root directory of the removable media. In this file, include the filename of the file to run, using the following syntax.

```
[autorun]
open = filename
```

Unless you specify a path, the system looks for the file in the root of the inserted media. If you want to run a file located in a subdirectory, include a path relative to the root; include that path with the file as in the following example.

```
open = My Directory\My File.exe
```

Running the file from a subdirectory does not change the current directory setting. The command-line string you supply can also include parameters or switches.

Because the autoplay feature is intended to provide automatic operation, design the file you specify in the `Autorun.inf` file to provide visual feedback quickly to confirm the successful insertion of the media. Consider using a startup up window with a graphic or animated sequence. If the process you are automating requires a long load time or requires user input, offer the user the option to cancel the process.

Although you can use this feature to install an application, avoid writing files to the user's local disk without the user's confirmation. Even when you get the user's confirmation, minimize the file storage requirements, particularly for CD-ROM games or educational applications. Consuming a large amount of local file space defeats some of the benefits of the turnkey operation that the autoplay feature provides. Also, because a network administrator or the user can disable this feature, avoid depending on it for any required operations.

You can define the icon that the system displays for the media by including an entry in the Autorun.inf file that includes the filename (and optionally the path) including the icon using the following form.

```
icon = filename
```

The filename can specify an icon, a bitmap, an executable, or a dynamic-link library file. If the file contains more than one icon resource, specify the resource with a number after the filename — for example, My File.exe, 1. The numbering follows the same conventions as the registry. The default path for the file will be relative to the Autorun.inf file. If you want to specify an absolute path for an icon, use the following form.

```
defaulticon = path
```

The system automatically provides a pop-up menu for the icon and includes AutoPlay as the default command on that menu, so that double-clicking the icon will run the Open = line. You can include additional commands on the menu for the icon by adding entries for them in the Autorun.inf file, using the following form.

```
shell\verb\command = filename  
shell\verb = Menu Item Name
```

To define an access key assignment for the command, precede the character with an ampersand (&). For example, to add the command Read Me First to the menu of the icon, include the following in the Autorun.inf file.

```
shell\readme\command = Notepad.exe My Directory\Readme.txt  
shell\readme = Read &Me First
```

Although AutoPlay is typically the default menu item, you can define a different command to be the default by including the following line.

```
shell = verb
```

When the user double-clicks on the icon, the command associated with this entry will be carried out.

System Naming Conventions

Windows provides support for filenames up to 255 characters long. Use the long filename when displaying the name of a file. Avoid displaying the filename extension unless the user chooses the option to display extensions or when the file type is not registered.

Because the system uses three-letter extensions to describe a file type, do not use extensions to distinguish different forms of the same file type. For example, if your application has a function that automatically backs up a file, name the backup file Backup of *filename.ext* (using its existing extension) or some reasonable equivalent, not *filename.bak*. The latter implies a change of the file's type. Similarly, do not use a Windows filename extension unless your file fits the type description.

Long filenames can include any character, except the following.

```
\\ / : * ? < > | " ' ,
```

When your application automatically supplies a filename, use a name that communicates information about its creation. For example, files created by a particular application should use either the application-supplied type name or the short type name as a proposed name — for example, worksheet or document. When that file exists already in the target directory, add a number to the end of the proposed name — for example, Document (2). When adding numbers to the end of a proposed filename, use the first number of an ordinal sequence that does not conflict with an existing name in that directory.



The system automatically formats a filename correctly if you use the **SHGetFileInfo** or **GetFileName** function. For more information about these functions, see the documentation included in the Win32 SDK.

When saving a file, make certain you preserve the creation date of the file. For simple applications that open and save a file, this happens automatically. However, more sophisticated applications may create temporary files, delete the original file, and rename the temporary file to the original filename. In this case, the application needs to copy the creation date as well from the old file to the new, using the standard system functions. Certain system file management functionality may depend on the correct creation date.

When you create a filename, the system automatically creates an MS-DOS filename (alias) for a file. The system displays both the long filename and the MS-DOS filename in the property sheet for the file.

When a file is copied, use the words “Copy of” as part of the generated filename — for example, “Copy of Sample” for a file named “Sample.” If the prefix “Copy of” is already assigned to a file, include a number in parentheses — for example, “Copy (2) of Sample”. You can apply the same naming scheme to links, except the prefix is “Link to” or “Shortcut to.”

It is also important to support UNC paths for identifying the location of files and folders. UNC paths and filenames have the following form.

\\Server\Share\Directory\Filename.ext

Using UNC names enables the user to directly browse the network and open files without having to make explicit network connections.

Wherever possible, display the full name of a file (without the extension). The number of characters you’ll be able to display depends somewhat on the font used and the context in which the name is displayed. In any case, supply enough characters such that the user can reasonably distinguish between names. Take into account common prefixes such as “Copy of” or “Shortcut to”. If you don’t display the full name, indicate that it has been truncated by appending an ellipsis to the end of the name.

You can use an ellipsis to abbreviate path names, in a displayable, but noneditable situation. In this case, include at least the first two entries of the beginning and the end of the path, using ellipses as notation for the names in between, as in the following example.

\\My Server\My Share\...\My Folder\My File

When using an icon to represent a network resource, label the icon with the name of the resource. If you need to show the network context rather than using a UNC path, label the resource using the following format.

Resource Name on Computer Name

Taskbar Integration

The system provides support for integrating your application's interface with the taskbar. The following sections provide information on some of the capabilities and appropriate guidelines.

Taskbar Window Buttons

When an application creates a primary window, the system automatically adds a taskbar button for that window and removes it when that window closes. For some specialized types of applications that run in the background, a primary window may not be necessary. In such cases, make certain you provide reasonable support for controlling the application using the commands available on the application's icon; it should not appear as an entry on the taskbar, however. Similarly, the secondary windows of an application should also not appear as a taskbar button.

The taskbar window buttons support drag and drop, but not in the conventional way. When the user drags an object over a taskbar window button, the system automatically restores the window. The user can then drop the object in the window.

Status Notification

The system allows you to add status or notification information to the taskbar. Because the taskbar is a shared resource, add information to it that is of a global nature only or that needs monitoring by the user while working with other applications.

Present status notification information in the form of a graphic supplied by your application, as shown in Figure 10.2.



The **Shell_NotifyIcon** function provides support for adding a status item in the taskbar. For more information about this function, see the documentation included in the Win32 SDK.



Figure 10.2 Status indicator in the taskbar

When adding a status indicator to the taskbar, also support the following interactions:

- Provide a pop-up window that displays further information or controls for the object represented by the status indicator when the user clicks with button 1. For example, the audio (speaker) status indicator displays a volume control. Use a pop-up window to supply for further information rather than a dialog box, because the user can dismiss the window by clicking elsewhere. Position the pop-up window near the status indicator so the user can navigate to it quickly and easily. Avoid displaying other types of secondary windows because they require explicit user interaction to dismiss them. If there is no information or control that applies, do not display anything.
- Display a pop-up menu for the object represented by the status indicator when the user clicks on the status indicator with button 2. On this menu, include commands that bring up property sheets or other windows related to the status indicator. For example, the audio status indicator provides commands that display the audio properties as well as the Volume Control mixer application.
- Carry out the default command defined in the pop-up menu for the status indicator when the user double-clicks.

- Display a tooltip that indicates what the status indicator represents. For example, this could include the name of the indicator, a value, or both.
- Provide the user an option to not display the status indicator, preferably in the property sheet for the object displaying the status indicator. This allows the user to determine which indicator to include in this shared space. You may need to provide an alternate means of conveying this status information when the user turns off the status indicator.

Message Notification

When your application's window is inactive but must display a message, rather than displaying a message box on top of the currently active window and switching the input focus, flash your application's title bar and taskbar window button to notify the user of the pending message. This avoids interfering with the user's current activity but lets the user know a message is waiting. When the user activates your application's window, the application can display a message box.

Use the system setting for the cursor blink rate for your flash rate. This allows the user to control the flash rate to a comfortable frequency.

Rather than flashing the button continually, you can flash the window button only a limited number of times (for example, three), then leave the button in the highlighted state, as shown in Figure 10.3. This lets the user know there is still a pending message.



Figure 10.3 Flashing a taskbar button to notify a user of a pending message

This cooperative means of notification is preferable unless a message relates to the system integrity of the user's data, in which case your application may immediately display a system modal message box. In such cases, flush the input queue so that the user does not inadvertently select a choice in that message box.

The **FlashWindow** function supports flashing your title bar and taskbar window button. For more information about this function, see the documentation included in the Win32 SDK.

The **GetCaretBlinkTime** function provides access to the current cursor blink rate setting. For more information about this function, see the documentation included in the Win32 SDK.

Application Desktop Toolbars

The system supports applications supplying their own desktop toolbars, also referred to as access bars or appbars, that operate similarly to the Windows taskbar. These may be docked to the edges of a screen and provide access to controls, such as buttons, for specific functions.

The system supports the same auto-hide behavior for application desktop toolbars as it does for the taskbar. This allows the desktop toolbar to only be visible when the user moves the pointer to the edge of the screen. The system also provides the “always on top” behavior used by the taskbar. When the user sets this property, the taskbar always appears on top (in the Z order) of any windows and also acts as a boundary for windows set to maximize to the display screen size.

Desktop toolbars can also be undocked and displayed as a palette window or redocked at a different edge of the screen. In the undocked, displayed as a palette window state, the toolbar no longer constrains other windows. However, if it supports the Always on Top property, it remains on top of other application windows.



For more information on the recommended behavior for undocking and redocking toolbars, see Chapter 7, “Menus, Controls, and Toolbars.”

Before designing a desktop toolbar, consider whether your application’s tasks really require one. Remember that a desktop toolbar will potentially affect the visible area for all applications. Only provide one for frequently used interfaces that can be applied across applications and always design it to be an optional interface, allowing the user to close it or otherwise configure it not to appear. You may also want to consider removing it when a specific application or applications are closed.

When creating your own desktop toolbar, model its behavior on the taskbar. Consider using the system’s notification of when the taskbar’s auto-hide or Always on Top property changes to apply a desktop toolbar you provide. If this does not fit your design, be certain to provide your own property sheet for setting these attributes for your desktop toolbar. Note that the system only supports auto-hide functionality for one desktop toolbar on each edge of the display. In addition, always provide a pop-up menu to access commands that apply to your desktop toolbar, such as Close, Move, Size, and Properties (but not the commands included on the desktop toolbar).

You can choose to display a desktop toolbar when the user runs a specific application, or by creating a separate application and including a shortcut icon to it in the system's Startup folder. Preferably set the initial size and position of your desktop toolbar so that it does not interfere with other desktop toolbars or the taskbar. However, the system does support multiple desktop toolbars to be docked along the same edge of the display screen. When docking on the same edge as the taskbar, the system places the taskbar on the outermost edge.

Your desktop toolbar can include any type of control. A desktop toolbar can also be a drag and drop target. Follow the recommendations outlined in this guide for supporting appropriate interaction.

Full-Screen Display

Although the taskbar and application desktop toolbars normally constrain or clip windows displayed on the screen, you can define a window to the full extent of the display screen. Because this is not the typical form of interaction, only consider using full-screen display for very special circumstances, such as a slide presentation, and only when the user explicitly chooses a command for this purpose. Make certain you provide an easy way for the user to return to normal display viewing. For example, you can display an on-screen button when the user moves the pointer that restores the display when the user clicks it. In addition, keyboard interfaces, like ALT+TAB and ESC, should automatically restore the display.

Remember that desktop toolbars, including the taskbar, should support auto-hide options that allow the user to configure them to reduce their visual impact on the screen. Consider whether this auto-hide capability may be sufficient before designing your application to require a full-screen presentation. Advising the user to close or hide desktop toolbars may provide you with sufficient space without having to use the full display screen.

Recycle Bin Integration



The Recycle Bin provides a repository for deleted files. If your application includes a facility for deleting files, support the Recycle Bin interface. You can also support deletion to the Recycle Bin for nonfile objects by first formatting the deleted data as a file by writing it to a temporary file and then calling the system functions that support the Recycle Bin.



The **SHFileOperation** function supports deletion using the Recycle Bin interface. For more information about this function, see the documentation included in the Win32 SDK.

Control Panel Integration



The Windows Control Panel includes special objects that let users configure aspects of the system. Your application can add Control Panel objects or add property pages to the property sheets of existing Control Panel objects.

Adding Control Panel Objects

You can create your own Control Panel objects. Most Control Panel objects supply only a single secondary window, typically a property sheet. Define your Control Panel object to represent a concrete object rather than an abstract idea.

Every Control Panel object is a dynamic-link library. To ensure that the dynamic-link library can be automatically loaded by the system, set the file's extension to .CPL and install it in the Windows System directory.



The system automatically caches information about Control Panel objects in order to provide quick user access, provided that the Control Panel object supports the correct system interfaces. For more information about developing Control Panel objects, see the documentation included in the Win32 SDK.

Adding to the Passwords Object

The Passwords object in Control Panel supplies a property sheet that allows the user to set security options and manage passwords for all password-protected services in the system. The Passwords object also allows you to add the name of a password-protected service to the object's list of services and use the Windows login password for all password-protected services in the system.

When you add your service to the Passwords object, the name of the service appears in the Select Password dialog box that appears when the user chooses Change Other Passwords. The user can then change the password for the service by selecting the name and filling in the resulting dialog box. The name of your service also appears in the Change Windows Password dialog box; the name appears with a check box next to it. By setting the check box option, the user chooses to keep the password for the service identical to the Windows login password. Similarly, the user can disassociate the service from the Windows login password by toggling the check box setting off.

To add your service to the Passwords object, register your service under the **HKEY_LOCAL_MACHINE** key.

HKEY_LOCAL_MACHINE

System

CurrentControlSet

Control

PwdProvider

Provider Name Value Name = Value

You can also add a page to the property sheet of the Passwords object to support other security-related services that the user can set as property values. Add a property page if your application provides security-related functionality beyond simple activation and changing of passwords. To add a property page, follow the conventions for adding shell extensions.



For more information about registering your password service, see the documentation included in the Win32 SDK.

Plug and Play Support

Plug and Play is a feature of Windows that, with little or no user intervention, automatically installs and configures drivers when their corresponding hardware peripherals are plugged into a PC. This feature applies to peripherals designed according to the Plug and Play specification. Supporting and appropriately adapting to Plug and Play hardware change can make your application easier to use. Following are some examples of supporting Plug and Play:

- Resizing your windows and toolbars relevant to screen size changes.
- Prompting users to shut down and save their data when the system issues a low power warning.
- Warning users about open network files when undocking their computers.
- Saving and closing files appropriately when users eject or remove removable media or storage devices or when network connections are broken.

System Settings and Notification

The system provides standard metrics and settings for user interface aspects, such as colors, fonts, border width, and drag rectangle (used to detect the start of a drag operation). The system also notifies running applications when its settings change. When your application starts up, query the system to set your application's user interface to match the system parameters to ensure visual and operational consistency. Also, design your application to adjust itself appropriately when the system notifies it of changes to these settings.



The **GetSystemMetrics**, **GetSysColor**, and **SystemParametersInfo** functions and the **WM_SETTINGSCCHANGE** message are important to consider when supporting standard system settings. For more information about these system interfaces, see the documentation included in the Win32 SDK.

Modeless Interaction

When designing your application, try to ensure that it is as interactive and nonmodal as possible. Here are some suggested ways of doing this:

- Use modeless secondary windows wherever possible.
- Segment processes, like printing, so you do not need to load the entire application to perform the operation.
- Make long processes run in the background, keeping the foreground interactive. For example, when something is printing, it should be possible to minimize the window even if the document cannot be altered. The multitasking support of Windows provides for defining separate processes, or *threads*, in the background.



For more information about threads, see the documentation included in the Win32 SDK.

Exhibit 0001A

**WINDOWS
INTERFACE
GUIDELINES FOR
SOFTWARE
DESIGN**

FT MEADE
GenColl

QA 76

.76

.W56 W553

1995

COPY 2

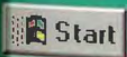


Microsoft® Professional Reference

Covers
Windows 95 &
Windows NT™!



The Windows® Interface Guidelines for Software Design



Microsoft Press



The Windows[®] Interface Guidelines for Software Design

Microsoft[®]

PUBLISHED BY

Microsoft Press

A Division of Microsoft Corporation

One Microsoft Way

Redmond, Washington 98052-6399

Copyright © 1995 by Microsoft Corporation

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of the publisher.

Library of Congress Cataloging-in-Publication Data

The Windows interface guidelines for software design.

p. cm.

Includes index.

ISBN 1-55615-679-0

1. Microsoft Windows (Computer file) 2. User interfaces (Computer systems) 3. Computer software--Development. I. Microsoft Corporation.

QA76.76.W56W553 1995

005.265--dc20

95-330

CIP

Printed and bound in the United States of America.

1 2 3 4 5 6 7 8 9 QE QE 0 9 8 7 6 5

Distributed to the book trade in Canada by Macmillan of Canada, a division of Canada Publishing Corporation.

A CIP catalogue record for this book is available from the British Library.

Microsoft Press books are available through booksellers and distributors worldwide. For further information about international editions, contact your local Microsoft Corporation office. Or contact Microsoft Press International directly at fax (206) 936-7329.

Information in this document is subject to change without notice and does not represent a commitment on the part of Microsoft Corporation. Companies, names, and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. The furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property rights.

Adobe, PostScript, and TIFF are trademarks of Adobe Systems, Inc. Apple and TrueType are registered trademarks of Apple Computer, Inc. Borland and Quattro are registered trademarks of Borland International, Inc. Frutiger is a registered trademark of Eltra Corporation. HP and LaserJet are registered trademarks of Hewlett-Packard Company. Backup was developed for Microsoft by Colorado Memory Systems, Inc., a division of Hewlett-Packard Company. HyperTerminal is a trademark of Hilgraeve, Inc. 1-2-3 and Lotus are registered trademarks of Lotus Development Corporation. Microsoft, Microsoft Press, Microsoft Press logo, MS, MS-DOS, PowerPoint, Visual Basic, Windows, Windows logo, and XENIX are registered trademarks and Windows NT is a trademark of Microsoft Corporation. Arial, Bodoni, Swing, and Times New Roman are registered trademarks of The Monotype Corporation PLC. Paintbrush is a trademark of Wordstar Atlanta Technology Center.

.....	xv
is Guide	xvi
he Guidelines	xvii
sed in This Guide	xviii

LS OF DESIGNING USER INTERACTION

nciples and Methodology	
Design Principles	3
rol	3
.....	4
.....	5
.....	6
.....	7
.....	7
.....	8
ology	9
Design Team	9
Cycle	9
essment in the Design Process	12
Users	15
fs	16

Chapter 2 Basic Concepts

Data-Centered Design	17
Objects as Metaphor	18
Object Characteristics	18
Relationships	19
Composition	19
Persistence	20
Putting Theory into Practice	20

Chapter 3 The Windows Environment

The Desktop	23
The Taskbar	24
The Start Button	25
Window Buttons	26
The Status Area	26
Icons	26
Windows	28

Chapter 4 Input Basics

Mouse Input	29
Mouse Pointers	29
Mouse Actions	31
Keyboard Input	32
Text Keys	33
Access Keys	33
Mode Keys	34
Shortcut Keys	35
Pen Input	37
Pen Pointers	39
Pen Gestures	40
Pen Recognition	41
Ink Input	41
Targeting	42

Chapter 5 General Interaction Techniques

Navigation	43
Mouse and Pen Navigation	44
Keyboard Navigation	44
Selection	45
Selection Feedback	46
Scope of Selection	47
Hierarchical Selection	47
Mouse Selection	48
Pen Selection	55
Keyboard Selection	56
Selection Shortcuts	57
Common Conventions for Supporting Operations	58
Operations for a Multiple Selection	58
Default Operations and Shortcut Techniques	59
View Operations	59
Editing Operations	62
Editing Text	62
Handles	63
Transactions	64
Properties	65
Pen-Specific Editing Techniques	66
Transfer Operations	72
Command Method	74
Direct Manipulation Method	77
Transfer Feedback	83
Specialized Transfer Commands	86
Shortcut Keys for Transfer Operations	87
Creation Operations	87
Copy Command	87
New Command	88
Insert Command	88
Using Controls	88
Using Templates	88
Operations on Linked Objects	89

PART II WINDOWS INTERFACE COMPONENTS

Chapter 6 Windows

Common Types of Windows	95
Primary Window Components	95
Window Frames	96
Title Bars	96
Title Bar Icons	97
Title Text	98
Title Bar Buttons	101
Basic Window Operations	103
Activating and Deactivating Windows	103
Opening and Closing Windows	104
Moving Windows	106
Resizing Windows	106
Scrolling Windows	109
Splitting Windows	116

Chapter 7 Menus, Controls, and Toolbars

Menus	121
The Menu Bar and Drop-down Menus	121
Common Drop-down Menus	124
Pop-up Menus	126
Pop-up Menu Interaction	128
Common Pop-up Menus	129
Cascading Menus	133
Menu Titles	134
Menu Items	135
Controls	140
Buttons	141
List Boxes	149
Text Fields	157
Other General Controls	163
Pen-Specific Controls	169
Toolbars and Status Bars	172
Interaction with Controls in Toolbars and Status Bars	173
Support for User Options	174
Toolbar and Status Bar Controls	175
Common Toolbar Buttons	176

Chapter 8 Secondary Windows

Characteristics of Secondary Windows	179
Appearance and Behavior	179
Window Placement	183
Modeless vs. Modal	183
Default Buttons	184
Navigation in Secondary Windows	185
Validation of Input	187
Property Sheets and Inspectors	187
Property Sheet Interface	188
Property Sheet Commands	190
Closing a Property Sheet	191
Property Inspectors	191
Properties of a Multiple Selection	192
Properties of a Heterogeneous Selection	193
Properties of Grouped Items	193
Dialog Boxes	193
Dialog Box Commands	194
Layout	194
Common Dialog Box Interfaces	195
Palette Windows	207
Message Boxes	209
Title Bar Text	209
Message Box Types	209
Command Buttons in Message Boxes	212
Message Box Text	213
Pop-up Windows	215

PART III DESIGN SPECIFICATIONS AND GUIDELINES

Chapter 9 Window Management

Single Document Window Interface	219
Multiple Document Interface	220
Opening and Closing MDI Windows	222
Moving and Sizing MDI Windows	223
Switching Between MDI Child Windows	225
MDI Alternatives	225
Workspaces	227
Workbooks	229
Projects	231
Selecting a Window Model	233
Presentation of Object or Task	233
Display Layout	234
Data-Centered Design	235
Combination of Alternatives	235

Chapter 10 Integrating with the System

The Registry	237
Registering Application State Information	238
Registering Application Path Information	241
Registering File Extensions	242
Supporting Creation	249
Registering Icons	250
Registering Commands	251
Enabling Printing	253
Registering OLE	253
Registering Shell Extensions	254
Supporting the Quick View Command	256
Registering Sound Events	257
Installation	257
Copying Files	257
Providing Access to Your Application	260
Designing Your Installation Program	260

Installing Fonts	262
Installing Your Application on a Network	262
Uninstalling Your Application	263
Supporting AutoPlay	264
System Naming Conventions	266
Taskbar Integration	268
Taskbar Window Buttons	268
Status Notification	269
Message Notification	270
Application Desktop Toolbars	271
Full-Screen Display	272
Recycle Bin Integration	273
Control Panel Integration	273
Adding Control Panel Objects	273
Adding to the Passwords Object	273
Plug and Play Support	275
System Settings and Notification	275
Modeless Interaction	276

Chapter 11 Working with OLE Embedded and OLE Linked Objects

The Interaction Model	277
Creating OLE Embedded and OLE Linked Objects	279
Transferring Objects	279
Inserting New Objects	285
Displaying Objects	290
Selecting Objects	293
Accessing Commands for Selected Objects	295
Activating Objects	297
Outside-in Activation	297
Inside-out Activation	297
Container Control of Activation	298
OLE Visual Editing of OLE Embedded Objects	300
The Active Hatched Border	304

Contents

Menu Integration	305
Keyboard Interface Integration	308
Toolbars, Frame Adornments, and Palette Windows	310
Opening OLE Embedded Objects	313
Editing an OLE Linked Object	316
Automatic and Manual Updating	318
Operations and Links	319
Types and Links	320
Link Management	320
Accessing Properties of OLE Objects	321
The Properties Command	321
The Links Command	324
Converting Types	326
Using Handles	329
Undo Operations for Active and Open Objects	330
Displaying Messages	332
Object Application Messages	332
OLE Linked Object Messages	334
Status Line Messages	336

Chapter 12 User Assistance

Contextual User Assistance	339
Context-Sensitive Help	339
Guidelines for Writing Context-Sensitive Help	342
Tooltips	343
Status Bar Messages	344
Guidelines for Writing Status Bar Messages	345
The Help Command Button	346
Task-Oriented Help	347
Task Topic Windows	347
Guidelines for Writing Task Help Topics	348
Shortcut Buttons	349

Reference Help	350
The Reference Help Window	351
Guidelines for Writing Reference Help	352
The Help Topics Browser	354
The Help Topics Tabs	354
Wizards	358
Guidelines for Designing Wizards	359
Guidelines for Writing Text for Wizard Pages	363

Chapter 13 Visual Design

Visual Communication	365
Composition and Organization	366
Color	368
Fonts	370
Dimensionality	371
Design of Visual Elements	372
Basic Border Styles	372
Window Border Style	373
Button Border Styles	374
Field Border Style	375
Status Field Border Style	376
Grouping Border Style	376
Visual States for Controls	377
Layout	384
Font and Size	384
Capitalization	387
Grouping and Spacing	388
Alignment	389
Placement	389
Design of Graphic Images	390
Icon Design	391
Pointer Design	394
Selection Appearance	395
Highlighting	396
Handles	397
Transfer Appearance	398
Open Appearance	399
Animation	400

Chapter 14 Special Design Considerations

Sound	401
Accessibility	403
Types of Disabilities	404
Types of Accessibility Aids	406
Compatibility with Screen Review Utilities	408
The User's Point of Focus	411
Timing and Navigational Interfaces	411
Color	412
Keyboard and Mouse Interface	413
Documentation, Packaging, and Support	414
Usability Testing	414
Internationalization	415
Text	416
Graphics	417
Keyboards	418
Character Sets	419
Formats	419
Layout	420
References to Unsupported Features	420
Network Computing	421
Leverage System Support	421
Client-Server Applications	421
Shared Data Files	422
Record Processing	422
Telephony	423
Microsoft Exchange	424
Coexisting with Other Information Services	424
Adding Menu Items and Toolbar Buttons	424
Supporting Connections	425
Installing Information Services	425

PART IV APPENDIXES

Appendix A Mouse Interface Summary

Interaction Guidelines for Common	
Unmodified Mouse Actions	429
Interaction Guidelines for Using the SHIFT Key	
to Modify Mouse Actions	431
Interaction Guidelines for Using the CTRL Key	
to Modify Mouse Actions	435

Appendix B Keyboard Interface Summary

Common Navigation Keys	437
Common Shortcut Keys	438
Windows Keys	439
Accessibility Keys	440
Access Key Assignments	441

Appendix C Guidelines Summary

General Design	443
Design Process	444
Input and Interaction	444
Windows	445
Control Usage	446
Integration	447
User Assistance	448
Visual Design	448
Sound	449
Accessibility	449
International Users	450
Network Users	450

Appendix D Supporting Specific Versions of Microsoft Windows

Microsoft Windows 3.1 451
Microsoft Windows NT 3.51 453

Appendix E International Word Lists

International Word Lists 455

Glossary 511
Bibliography 523
Index 527

Introduction

These guidelines supersede those issued for Windows version 3.1 and all previous releases and are specific to the development of applications designed for Microsoft® Windows®, Microsoft® Windows NT™ Workstation, and Microsoft® Windows NT Server. There is no direct relationship between these guidelines and those provided for other operating systems.

For more information about special considerations concerning developing applications for both Windows 95 and Windows NT operating system, see Appendix D, “Supporting Specific Versions of Windows.”

Conventions Used in This Guide

The following conventions are used throughout this guide.

Convention	Indicates
	A reference to related topics in this guide or other books that provide more information about the topic.
	Additional or special information about the topic.
SMALL CAPITAL LETTERS	Names of keys on the keyboard — for example, SHIFT, CTRL, or ALT.
KEY+KEY	Key combinations for which the user must press and hold down one key and then press another — for example, CTRL+P or ALT+F4.
<i>Italic text</i>	New terms and variable expressions, such as parameters.
Bold text	Win32 API keywords and registry key entries.
Registry text	Examples of registry entries.
[]	Optional information.

Index

Windows (*continued*)

- full-screen display 272
 - illustrations
 - common window components 96
 - MDI parent and child window title bars 100
 - moving the split bar 118
 - scroll bar 110, 112
 - scrolling with the scroll bar shaft 114
 - split box location 117
 - split window 116
 - title bar icons 97–98
 - title text order 99
 - maximizing 107, 223–224
 - minimizing 108, 223–224
 - moving 106, 223–224
 - opening 60, 104, 222
 - palette windows 207–208, 310–313
 - pop-up menu for 129–130
 - pop-up windows 215
 - primary 95
 - property inspectors 187, 191–192
 - property sheets 187–193
 - restoring 108
 - scrolling 109–116
 - size grip for 109
 - size recommendations 386
 - sizing 106–109, 223
 - splitting 116–119
 - summary of guidelines 445
 - switching between 103, 181, 225
 - use of windows, described 28
- Windows 3.1–based applications, porting 21
- Windows environment
- See also* Integrating with the system
 - overview of 23–28

Windows keys

- avoiding use with modifier keys 36
 - table of WINDOWS key combinations 439
- Windows NT, supporting specific versions of 453–454
- Windows version 3.1, supporting 451–452
- Wizards 358–363
- Word lists for localization 455–510
- Word wrap 62
- Workbooks 229–231
- Workspace menu 306
- Workspaces 227–229
- Writing tool for pen interface 66–71

Z

Z order

- activating windows 103
- for application desktop toolbars 271
- for palette windows 208
- for secondary windows 181
- opening windows 60, 105
- pasting objects in Z-ordered contexts 76
- selecting objects in Z-ordered contexts 54

9 284 96

THE WINDOWS® INTERFACE GUIDELINES FOR SOFTWARE DESIGN

Here are the Microsoft® guidelines for creating well-designed, visually and functionally consistent user interfaces for applications that run on the Microsoft Windows operating system. **THE WINDOWS INTERFACE GUIDELINES FOR SOFTWARE DESIGN** is an essential handbook for all programmers and designers working with the latest release of Windows, regardless of experience level or development tools used.

This book covers the basic principles of user interface design and design methodologies and specifies how data-centered concepts such as objects and properties can be applied to interface design. It includes detailed information on mouse, keyboard, and pen interaction and on how to use the common interface elements supplied by the system. This book also includes information about special considerations such as network computing, international users, and users with disabilities.

Topics include:

- *General input techniques*—navigation, selection, viewing, editing, and creation, including both command and direct manipulation methods such as drag and drop.
- *Windows*—primary and secondary types and their components, including property sheets, dialog boxes, message boxes, palette windows, and pop-up windows.
- *Menus, controls, and toolbars*—types and their components and when to use them.
- *Microsoft OLE*—how to support design interfaces for OLE embedded and linked objects, visual editing, and other forms of activation.
- *User assistance*—how to use contextual forms of help, including tooltips and wizards.
- *Integration with the system*—designing your software so that its interface functions and operates consistently with Windows.
- *Visual design*—effective use of color, layout, fonts, and graphics.

If you are developing or are considering developing applications for Microsoft Windows,

THE WINDOWS INTERFACE GUIDELINES FOR SOFTWARE DESIGN is a key resource.

U.S.A. **\$29.95**

U.K. **£27.49**

Canada **\$39.95**

[Recommended]

Microsoft Press

Operating Systems/Windows/Programming

ISBN 1-55615-679-0



9 781556 156793





Exhibit 0001B

← → ↻ <https://search.catalog.loc.gov/instances/34938f56-c4a3-5eca-9fc3-b75d39d98f89?option=lccn&query=95000330> ☆

LIBRARY LIBRARY OF CONGRESS

[Catalog home](#) [My dashboard](#) [Browse](#) [Library resources](#)

LCCN 95000330 [Advanced search](#)

[New search](#)

Book

The Windows interface guidelines for software design.

Contributors: [Microsoft Press](#)

Published: Microsoft Press, Redmond, WA, c1995

✓ **Available:** 1 of 2 copies remaining

[Request item](#) [Item availability](#)



Full record

MARC record

Additional links

Permalink

<https://lccn.loc.gov/95000330> 

Look for online version

 [Find it](#)

Publication details

Language

English

Publisher

Microsoft Press, Redmond, WA

Library of Congress subjects

[Microsoft Windows \(Computer file\)](#)

[User interfaces \(Computer systems\)](#)

[Computer software--Development](#)

Document type

Book

Physical description

xx, 556 p. : ill. (some col.) ; 23 cm.

Contents/Summary**Notes**

"Microsoft professional reference"--Cover

"Covers Windows 95 & Windows NT"--Cover

Bibliographic notes

Includes bibliographical references (p. [523]-525) and index

Identifiers and numbering**ISBN**

9781556156793

1556156790

LCCN

95000330

Classification - LC

QA76.76.W56 W553 1995

Classification - Dewey

005.265

 [Please contact library staff for additional help](#)

Availability & locations

Items: 1-2 of 2

Status	Location	Call number
 In Transit	Offsite General Collections Offsite	QA76.76.W56 W553 1995 FT MEADE
 Available	Offsite General Collections Offsite	QA76.76.W56 W553 1995 LANDOVR

[Ask a librarian](#)[Feedback](#)[Contact us](#)

Full record

MARC record

Staff view

LEADER 01466pam a2200373 a 4500

001 3977115

005 20250607161019.0

008 950515s1995 waua b 001 0 eng

035 \$a 3977115

906 \$a 7 \$b cbc \$c orignew \$d 1 \$e ocip \$f 19 \$g y-gencatlg

955 \$a pc17 to ja00 05-15-95; jf06 to subj 05-16-95; jf04 to S.L. 05-16-95; jf12 05-16-95; CIP ver. pv08 09-13-95

955 \$a LCAP batch update 2025-06-02-04:00: LCAPM-644, LCAPM-150

010 \$a 95000330

020 \$a 1556156790

020 \$a 9781556156793

020	\$a 1556156790
020	\$a 9781556156793
035	\$9 (DLC) 95000330
040	\$a DLC \$c DLC \$d DLC
050 0 0	\$a QA76.76.W56 \$b W553 1995
082 0 0	\$a 005.265 \$2 20
245 0 4	\$a The Windows interface guidelines for software design.
260	\$a Redmond, WA : \$b Microsoft Press, \$c c1995.
300	\$a xx, 556 p. : \$b ill. (some col.) ; \$c 23 cm.
336	\$a text \$b txt \$2 rdacontent
337	\$a unmediated \$b n \$2 rdamedia
338	\$a volume \$b nc \$2 rdacarrier
500	\$a "Microsoft professional reference"--Cover.
500	\$a "Covers Windows 95 & Windows NT"--Cover.

504	\$a Includes bibliographical references (p. [523]-525) and index.
630 0 0	\$a Microsoft Windows (Computer file)
650 0	\$a User interfaces (Computer systems)
650 0	\$a Computer software \$x Development.
710 2	\$a Microsoft Press.
991	\$b c-GenColl \$h QA76.76.W56 \$i W553 1995 \$p 00034796804 \$t Copy 1 \$w BOOKS
999 f f	\$i 34938f56-c4a3-5eca-9fc3-b75d39d98f89 \$s 68acfeaa-882c-4730-b2eb-64ca066cbd2b

 [Please contact library staff for additional help](#)

Availability & locations

Items: 1-2 of 2

Status	Location	Call number
 In Transit	Offsite General Collections Offsite	QA76.76.W56 W553 1995 FT MEADE
 Available	Offsite General Collections Offsite	QA76.76.W56 W553 1995 LANDOVR



[Ask a librarian](#) | [Feedback](#) | [Contact us](#)

[Speech Enabled](#) | [Accessibility](#) | [Legal](#) | [Inspector general](#) | [External link disclaimer](#) | [USA.gov](#)

[EBSCO Help](#) | [Privacy policy](#) | [Terms of use](#)

© 2025 EBSCO Industries, Inc. All rights reserved
Version 2.13.0.release_v2.13.0_2025_08_09_21_06_55

Exhibit 0001C



LOCIS: Library of Congress Information System

TELNET locis.loc.gov

No login required

LOCIS: LIBRARY OF CONGRESS INFORMATION SYSTEM

- | | | | |
|----|---|---|-------------------|
| 1 | Library of Congress Catalog | 4 | Braille and Audio |
| 2 | Federal Legislation | 5 | Organizations |
| 3 | Copyright Information | 6 | Foreign Law |
| 7 | Searching Hours and Basics | | |
| 8 | Documentation and Classes | | |
| 9 | Library of Congress General Information | | |
| 12 | Comments and Logoff | | |

LIBRARY OF CONGRESS CATALOG

CHOICE

- | | |
|---|--|
| 1 | BOOKS: English language books 1968-, French 1973-, German, Portuguese, Spanish 1975-, other European languages 1976-77, non-European languages 1978-79. Some microforms 1984-. |
| 2 | BOOKS earlier than the dates above. Some serials, maps, music, audiovisual items. |

FILE

LOCI

PREM

The Wayback Machine - <https://web.archive.org/web/19970524160715/http://library.usask.ca:80/hytelnet/us3/us373.html>

LOCIS: Library of Congress Information System

TELNET locis.loc.gov

No login required

L O C I S : LIBRARY OF CONGRESS INFORMATION SYSTEM

- 1

Library of Congress Catalog

2

Federal Legislation

3

Copyright Information

7

Searching Hours and Basics

8

Documentation and Classes

9

Library of Congress General Information

12

Comments and Logoff
- 4

Braille and Audio

5

Organizations

6

Foreign Law

LIBRARY OF CONGRESS CATALOG	
CHOICE	FILE
1 BOOKS: English language books 1968-, French 1973-, German, Portuguese, Spanish 1975-, other European languages 1976-77, non-European languages 1978-79. Some microforms 1984-.	LOCI
2 BOOKS earlier than the dates above. Some serials, maps, music, audiovisual items.	PREM
3 Combination of files 1 and 2 above (LOCI and PREM).	
4 SERIALS cataloged at LC & some other libraries since 1973.	LOCS
5 MAPS and other cartographic items cataloged at LC 1968- and some other research libraries 1985-. Atlases (which are books) are in LOCI and PREM.	LOCM
6 SUBJECT TERMS and cross references from LC Subject Headings.	LCXR
12 Return to LOCIS MENU screen.	

Exhibit 0001D

libraries.org

A directory of libraries throughout the world

- Home »
- Libraries »
- Guides
- Documents
- Vendors »
- Products »
- News
- Procurement
- Member Login »

search: University of Minnesi

Library of Congress

United States -- Library of Congress

Address: 101 Independence Avenue SE
Washington, District of
Columbia
20540
United States

Phone: 202-707-5000

Connect to: [Library Web Site](#) ✓ [Online Catalog](#)

Library details: Library of Congress is a National library. This library is affiliated with [Library of Congress \(view map\)](#) . The collection of the library contains 175,000,000 volumes. The library circulates 250,000 items per year.

Other Info: Went live with Voyager on August 15, 1999. Currently planning to implement the Library Collections Access Platform (LCAP) on FOLIO and EBSCO discovery services in FY25.

Permalink:  <https://librarytechnology.org/library/2263>
(Use this link to refer back to this listing.)

Administration: The director of the library is Dr. Carla Hayden .

Organizational structure: The library is part of a publicly-funded organization. It operates on a Non-profit financial model.



Affiliations: The library is a member of the Association of Research Libraries

Collection size	175,000,000	volumes
Annual Circulation	250,000	transactions

 Wireless: The library offers wireless access to the Internet. <http://www.loc.gov/rr/wirelessaccess.html>

Technology Profile



Library of Congress



map location ✓

Related Libraries

- All members of the Association of Research Libraries members
- Directory of ARL Member Libraries
- Libraries located in Washington (District of Columbia)
- all National libs in District of Columbia

About libraries.org

libraries.org is a free directory of libraries throughout the world.

This directory is a component of Library Technology Guides.

Please contact Marshall Breeding to report

	Product Name	Year Contracted
Current Automation System	FOLIO -- EBSCO Information Services	2023
Previous Automation System	Voyager	1998
Discovery Service (w/index)	Ex Libris CDI	
Electronic Resource Management	Electronic Resource Management	
Item ID Type	Barcode	
This library is responsible for the procurement of the library automation system.		

corrections about information listed for this library.

[View Privacy and Security Report](#)

Identifiers	
libraries.org ID	2263
OCLC Symbol	DLC
WorldCat Registry ID	45266

Record History: This listing was created on Aug 9, 1998 and was last modified on Apr 10, 2025.

Updates: Corrections or Updates? Registered members of Library Technology Guides can submit updates to library listings in libraries.org. [Registration](#) is free and easy. Already registered? [login](#). Or, you can report corrections just by sending a message to [Marshall Breeding](#).

[libraries.org](#)

[Update this Entry](#)

[Library Technology Guides](#)

Maintained by [Marshall Breeding](#)

[terms of service](#)

Copyright 1994-2025

Exhibit 0001E

Copyright

Registration Number / Date:

TX0004182723 / 1995-12-15

Type of Work:

Text

Title:

The Windows interface guidelines for software design.

Date of Creation:

1995

Date of Publication:

1995-08-01

Copyright Claimant:

Microsoft Press a division of Microsoft Corporation

Authorship on Application:

text: Microsoft Corporation, employer for hire.

Description:

556 p.

Imprint:

Redmond, WA : Microsoft Press, c1995.

Names:

Microsoft Press

Microsoft Corporation

Microsoft Corporation. Microsoft Press

USCO Catalog Link:

https://publicrecords.copyright.gov/detailed-record/voyager_15160964

Disclaimer: This material was generated by the U.S. Copyright Office's Copyright Public Records System (CPRS). For certified records, contact the [Records Research and Certification Division](#). For information on searching copyright records, see [How to Investigate the Copyright Status of a Work \(Circular 22\)](#). For information on removing personal information from Copyright Office public records, refer to [Privacy: Public Copyright Registration Records\(Circular 18\)](#).

Exhibit 0001F

▲ Due to construction some items in the Buhr facility are unavailable, but may be requested through [Interlibrary Loan](#).



SEARCH

[Account](#) [Log in](#)[ANN ARBOR](#) [FLINT](#)

ISBN/ISSN/OCLC/etc

32590068

[Advanced](#)

① **Search Tip:** Search by ISSN (8-digit code), ISBN (13 or 10-digit code), or OCLC number (e.g., 0040-781X; 0747581088; 921446069).

[Everything](#) [Catalog](#) [Articles](#) [Databases](#) [Online Journals](#) [Guides and More](#)[Catalog](#) / [Record](#) **Book**

The Windows interface guidelines for software design. ☐

Record info:

Format	Book
Contributors	Microsoft Press. Browse in author list
Published/Created	Redmond, WA : Microsoft Press, c1995.
Language	English
Note	"Microsoft professional reference"--Cover. "Covers Windows 95 & Windows NT"--Cover.
Physical Description	xx, 556 p. : ill. (some col.) ; 23 cm
Bibliography	Includes bibliographical references (p. [523]-525) and index.
ISBN	1556156790
Call Number	QA 76.76 .W56 W5531 1995 Browse in call number list
OCLC Number	32590068
Subjects (LCSH)	Microsoft Windows (Computer file) Browse in subject list User interfaces (Computer systems) Browse Computer software -- Development Browse

Ask a Librarian ^

Academic Discipline

[Engineering](#) > [Computer Science](#)

The University of Michigan Library aims to describe its collections in a way that respects the people and communities who create, use, and are represented in them. We encourage you to [contact us](#) anonymously if you encounter harmful or problematic language in catalog records or finding aids. More information about our policies and practices is available at [Remediation of Harmful Language](#).

Available at:

Offsite Shelving · 1 item

Action	Description	Status	Call Number
Get This		On shelf	QA 76.76 .W56 W5531 1995

Shelf browse

[Browse in call number list](#)

Return to current record

[OLE 2.0 and DDE distilled : a...](#)

Williams, Al,
1963-

1994

QA 76.76 .W56
W4941 1994

[Microsoft Windows 3 developer's...](#)

Wilton, Richard,
1953-

1991

QA 76.76 .W56
W51 1991

CURRENT RECORD

[The Windows interface guidelines...](#)

1995

QA 76.76 .W56 W5531
1995

[Windows magazine.](#)

1991-1999

QA76.76.W56
W555

[Windows 2000 configuration wizards /...](#)

Syngress
Media, Inc.
Staff Corporat...

2000

QA76.76.W56
W563 2000

Actions Select what to do with this record.

<https://search.lib.umich.edu/catalog/record/990030260390106381?query=isbn:32590068>

2/5

Nintendo Exhibit 1007
Page 00186



Email



Text



Citation

Export Citation
(EndNote)Export Citation
(RIS)Copy
link

Date Last Indexed: 20250813

MARC Data

LDR	01489nam a2200373 a 4500
005	20241218083621.0
008	960215s1995 waua b 001 0 eng
001	990030260390106381
010	a 95000330
015	a GB95-95487
020	a 1556156790
035	a (MiU)003026039MIU01
035	a (0CoLC)ocm32590068
040	a DLC c DLC d UKM d EYM d UtOrBLW
050 0 0	a QA76.76.W56 b W553 1995
082 0 0	a 005.265 2 20
245 0 4	a The Windows interface guidelines for software design.
260	a Redmond, WA : b Microsoft Press, c c1995.
300	a xx, 556 p. : b ill. (some col.) ; c 23 cm
336	a text b txt 2 rdacontent
337	a unmediated b n 2 rdamedia

338		a volume b nc 2 rdacarrier
500		a "Microsoft professional reference"--Cover.
500		a "Covers Windows 95 & Windows NT"--Cover.
504		a Includes bibliographical references (p. [523]-525) and index.
630	0 0	a Microsoft Windows (Computer file) 0 http://id.loc.gov/authorities/names/n88027331 1 http://id.loc.gov/rwo/agents/n88027331
650	0	a User interfaces (Computer systems) 0 http://id.loc.gov/authorities/subjects/sh88001679
650	0	a Computer software x Development. 0 http://id.loc.gov/authorities/subjects/sh85029535
710	2	a Microsoft Press. 0 http://id.loc.gov/authorities/names/n86038786 1 http://id.loc.gov/rwo/agents/n86038786 0 http://viaf.org/viaf/157186296
908		a AuthComplete 2024-12-17
958		a MiU
959		a (notis)ULAQY5884
995		a 20
998		c aut15
BIB		u 2025-01-02 09:33:09 US/Eastern c 2021-06-21 19:09:55 US/Eastern s false
852	0	b OFFS a MiU c STATE h QA 76.76 .W56 W5531 1995 8 22987474300006381
974		8 22987474300006381 f 1 c STATE m BOOK a 39015034541212 e STATE u mdp.39015034541212 ic 20210109 7 23987474290006381 r 1996-02-16 05:59:00 US/Eastern h QA 76.76 .W56 W5531 1995 d OFFS b OFFS

[Give feedback about this page](#) 

[Home](#)

[Accessibility](#)

[Tips for Using Library Search](#)

[Get research help](#)

[About Library Search](#)

[Make an I.L.L. Request](#)

©2025 Regents of the University of Michigan. For details and exceptions, see the [Copyright Policy](#).

Exhibit 0001G

libraries.org

A directory of libraries throughout the world

- [Home »](#)
- [Libraries »](#)
- [Guides](#)
- [Documents](#)
- [Vendors »](#)
- [Products »](#)
- [News](#)
- [Procurement](#)
- [Member Login »](#)

found 6 items where the item contains **University of Michigan**. Showing item 1 of 6.

list

next >>

search:

University of Michigan Library

University of Michigan

Address: 913 South University Avenue
[Ann Arbor, Michigan](#)
48109-1190
United States

County: [Washtenaw](#) 

Connect to: [Library Web Site](#) ✓ [Online Catalog](#)

Library details: **University of Michigan Library** is an **Academic** library.
This library is affiliated with [University of Michigan](#) ([view map](#)) . The collection of the library contains **14,032,199** volumes. The library circulates **324,150** items per year. The library serves a population of **51,823** FTE Enrollment.

Other Info: Member of the Big Ten Academic Alliance

Permalink:  <https://librarytechnology.org/library/2089>
(Use this link to refer back to this listing.)

Administration: The director of the library is **Lisa R. Carter** .

Organizational structure: The library is part of a **publicly-funded** organization. It operates on a **Non-profit** financial model.

Academic Level: The library Serves an educational institution at the **University** level. The Carnegie Foundation designates this institution as: **Doctoral Universities: Very High Research Activity** (level 15).



Affiliations: The library is a member of the [Association of Research Libraries](#)

Statistics Academic			
Enrollment	51,823	FTE Students	
Collection size	14,032,199	volumes	
Annual Circulation	324,150	transactions	



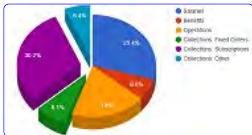
Harlan Hatcher Graduate Library



map location ✓

Related Libraries

- Other libraries with Carnegie class: **Doctoral Universities: Very High Research Activity**
- All members of the [Association of Research Libraries](#) members
- [Directory of ARL Member Libraries](#)
- Libraries located in [Ann Arbor](#) (Michigan)
- Libraries located in [Washtenaw](#) county (Michigan)
- [View map of libraries in Washtenaw County](#)
- [all Academic libs in Michigan](#)



[View Detailed Statistics](#)

Technology Profile

	Product Name	Year Contracted
Current Automation System	Alma	2019
Previous Automation System	ALEPH 500	2002
Previous Automation System	NOTIS	1987
Previous Automation System	GLIS	1980
Discovery Service (w/index)	Ex Libris CDI	2010
Discovery Interface	Locally Developed	2018
Previous Discovery Interface	VuFind	2009
Authentication Service	EZproxy	1990
Digital Collection Management	DLXS	1996
Institutional Repository	DSpace	2006
Digital Preservation	None	
Interlibrary Loan or Resource Sharing	Resource Sharing for Groups	
Resource Sharing Networks	Big Ten Academic Alliance	
Web Content Management	Drupal	
Item ID Type	Barcode	

This library is responsible for the procurement of the library automation system.

[View Privacy and Security Report](#)

Identifiers

libraries.org ID	2089
OCLC Symbol	EYM
WorldCat Registry ID	5293
NCES LIBID	170976
Ringgold ID	1259

Record History: This listing was created on Nov 30, 1997 and was last modified on Apr 11, 2025.

Updates: Corrections or Updates? Registered members of Library Technology Guides can submit updates to library listings in libraries.org. [Registration](#) is free and easy. Already registered? [login](#). Or, you can report corrections just by sending a message to [Marshall Breeding](#).

[libraries.org](#)

[Update this Entry](#)

[Library Technology Guides](#)

Maintained by [Marshall Breeding](#)

[terms of service](#)

Copyright 1994-2025

About libraries.org

libraries.org is a free directory of libraries throughout the world.

This directory is a component of [Library Technology Guides](#).

Please [contact Marshall Breeding](#) to report corrections about information listed for this library.

Exhibit 0001H

← → ↻ <https://web.archive.org/web/20000422051029/http://www1.fatbrain.com/asp/bookinfo/bookinfo.asp?theisbn=1556156790> 120% ☆

<http://www1.fatbrain.com/asp/bookinfo/bookinfo.asp?theisbn=1556156790> Go MAR APR JUN 1999 22 2000 2001 About this capture

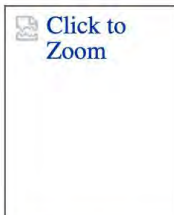
4 captures
22 Apr 2000 - 16 Jun 2000

fatbrain.com {★}
Because great minds think a lot

COMPUTING & INTERNET BUSINESS SCIENCE & ENGINEERING MEDICINE & BIOTECH GENERAL INTEREST eMatter @ mightywords

search Subjects New Titles Bestsellers Recommended Training & Certification What's Hot

QuickSearch Go! [Help Desk](#) [Your Account](#) [Shopping Basket](#)



The Windows Interface Guidelines for Software Design

By [Microsoft Corporation](#)

Online Price:

\$23.95 Softcover
20% off list (you save \$6.00)

List Price: \$29.95
556 Pages
Published by Microsoft Press
Date Published: 07/1995
ISBN: 1556156790

Add Item to Basket

In Stock - Orders received by 4 p.m. ship the [same day](#).

Summary

This well-organized and clearly written book provides guidelines for designing visually and functionally consistent user interfaces for Windows-based programs. It is the official book on Microsoft's user-interface design and can be read as a program specification for application developers who want to save training time, boost productivity, and promote user confidence in their applications.

Pages:

User Reviews

[The People Say...](#)



[You Think?](#)

[What Do](#)

More Information

[Description](#)

[Excerpt](#)

[Summary](#)

The Wayback Machine - https://web.archive.org/web/20000422051029/http://www1.fatbrain.com:80/asp/bookinfo/bookin...

fatbrain.com { }
Because great minds think a lot

COMPUTING & INTERNET BUSINESS SCIENCE & ENGINEERING MEDICINE & BIOTECH GENERAL INTEREST eMatter @ mightywords

search Subjects New Titles Bestsellers Recommended Training & Certification What's Hot

QuickSearch **Go!** [Help Desk](#) [Your Account](#) [Shopping Basket](#)



The Windows Interface Guidelines for Software Design

By [Microsoft Corporation](#)

Online Price:

\$23.95 Softcover

20% off list (you save \$6.00)

List Price: \$29.95

556 Pages

Published by Microsoft Press

Date Published: 07/1995

ISBN: 1556156790

[Add Item to Basket](#)

In Stock - Orders received by 4 p.m. ship the [same day](#)

User Reviews

[The People Say...](#)



[You Think?](#)

[What Do](#)

More Information

[Description](#)

[Excerpt](#)

[Summary](#)

[Our Review](#)

[Table of Contents](#)

[Brief](#)

Summary

This well-organized and clearly written book provides guidelines for designing visually and functionally consistent user interfaces for Windows-based programs. It is the official book on Microsoft's user-interface design and can be read as a program specification for application developers who want to save training time, boost productivity, and promote user confidence in their applications.

Pages:

Other Links

[Tell a Friend](#)

[Shipping options](#)

[Return to previous page](#)

Other people who bought this item also purchased:

[The C Programming Language, Second Edition](#)

[Programming Perl, Second Edition](#)

[Object-Oriented Technology for Real-Time Systems : A Practical Approach Using OMT and Fusion](#)

Click on this item's subject categories to see related items:

[Operating Systems : Windows / Windows '95 & '98 : Programming](#)

[Programming : User Interface : Windows](#)

[Programming : Platform-Specific : Windows / Windows '95 & '98](#)

[Security/Privacy Guarantee](#)
[Shipping Options](#)

[About Us](#)
[Keep Me Posted](#)
[Fatbrain Merchandise](#)

[Corporate Discounts](#)
[Our Partners](#)
[Affiliate Program](#)

[Retail Locations](#)
[Events](#)

[Home](#) | [Search](#) | [Subjects](#) | [New Titles](#)
[Bestsellers](#) | [Recommended](#) | [Training & Certification](#) | [What's Hot](#)
[Shopping Basket](#) | [Your Account](#) | [Help Desk](#)

Suggestions?



contact-us@fatbrain.com

Customer Service: (Toll-free) 800-789-8590

or 408-845-0402

[Copyright](#) © 1996 - 2000 Fatbrain.com, Inc.

QuickSearch:



Exhibit 0001I

Ex. 0001I

Citations to *Windows Interface*

Source Document:

Windows Interface Guidelines for Software Design. Microsoft Press. **Ex. 0001**. (“*Windows Interface*”)

Citations: 1996 to 1997 (in chronological order)

Dayton, T., Kramer, J., McFarland, A., & Heidelberg, M. (1996, April). Participatory GUI design from task models. In Conference Companion on Human Factors in Computing Systems (CHI '96). Association for Computing Machinery, New York, NY, USA, 375–376. <https://doi.org/10.1145/257089.257381>

Gale, S. (1996, April). A collaborative approach to developing style guides. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '96). Association for Computing Machinery, New York, NY, USA, 362–367. <https://doi.org/10.1145/238386.238572>

Chen, L. S., Liu, P. W., Chang, K. Y., Chen, J. P., Chen, S. C., Hong, H. C., & Liu, J. (1996, May). Using Hypermedia Computer-Aided Instruction. *IEEE Comput. Graph. Appl.* 16, 3 (May 1996), 52–57. <https://doi.org/10.1109/38.491185>

Jean-Daniel Fekete, J. D., & Beaudouin-Lafon, M. (1996, November). Using the multi-layer model for building interactive graphical applications. In Proceedings of the 9th annual ACM symposium on User interface software and technology (UIST '96). Association for Computing Machinery, New York, NY, USA, 109–118. <https://doi.org/10.1145/237091.237108>

Pandit, M. S., & Kalbag, S. (1997, January). The selection recognition agent: instant access to relevant information and operations. In Proceedings of the 2nd international conference on Intelligent user interfaces (IUI '97). Association for Computing Machinery, New York, NY, USA, 47–52. <https://doi.org/10.1145/238218.238285>

Tullis, T. S., & Pons, A. (1997, March). Designating required vs. optional input fields. In CHI '97 Extended Abstracts on Human Factors in Computing Systems (CHI EA '97). Association for Computing Machinery, New York, NY, USA, 259–260. <https://doi.org/10.1145/1120212.1120381>

Phelps, L. (1997, October). Active documentation: wizards as a medium for meeting user needs. In Proceedings of the 15th annual international conference on Computer documentation (SIGDOC '97). Association for Computing Machinery, New York, NY, USA, 207–210. <https://doi.org/10.1145/263367.263394>

Exhibit 0002



The following paper was originally published in the
Proceedings of the Large Installation System Administration of Windows NT Conference
Seattle, Washington, August 5–8, 1998

A Comparison of Large-Scale Software Installation Methods on NT and UNIX

Michail Gomberg, Rémy Evard, & Craig Stacey
Argonne National Laboratory

For more information about USENIX Association contact:

1. Phone: 510 528-8649
2. FAX: 510 548-5738
3. Email: office@usenix.org
4. WWW URL: <http://www.usenix.org/>

A Comparison of Large-Scale Software Installation Methods on NT and UNIX

Michail Gomberg, Rémy Evard, & Craig Stacey

Mathematics and Computer Science Division, Argonne National Laboratory

Abstract

Our computing environment consists of hundreds of UNIX and NT-based computers. We have a coherent UNIX software installation model that scales comfortably to hundreds of machines. We have spent a great deal of time in the last year learning to understand the software installation and support mechanisms for a similarly large NT infrastructure. In this paper, we examine the underlying requirements for large-scale software installation support, and compare and contrast the NT and UNIX environments, identifying strong points, weak points, and issues.

1. Introduction

The computing environment in the Mathematics and Computer Science Division of Argonne National Laboratory consists of nearly a thousand computers, including supercomputers, servers, workstations, desktop machines, and laptops. The majority of these are running UNIX, but the number of NT machines is steadily growing, particularly on desktops.

This set of machines is managed by a group of seven system administrators, meaning that, like most systems administration groups, we have more than enough work to go around. Our ability to administer the environment relies on scalable techniques. We have to manage large sets of machine as if they were one single system, most preferably from one single location. We were originally nervous about our ability to do this in a growing Windows NT environment due to its reputation as a system requiring hands-on administration. Software installation was one such area that we were worried about.

On each of our UNIX architectures, including SunOS, Solaris, Linux, FreeBSD, AIX, and IRIX installations, we use a common method of installing software. It scales well, in that we are able to install a given piece of software once and have it automatically work on every machine of a given type. This system is described in detail below.

As we began to look at software installation on NT in detail, we began to feel that our fears were justified. The usual software installation methods on NT require the administrator to sit in front of an individual machine, answer questions interactively, wait for the software to load, possibly reboot the machine, and then do a lot of tweaking in order to make the software available to other people who might log in to the machine. This approach doesn't scale to hundreds of machines. Fortunately, there are ways around most of these problems, but, as described below, they are not uniformly applicable to all NT applications. As a result, we started a long-term project to understand NT application installation better, to test the various installation tools, and to develop a philosophy and method of software installation that scaled as well as the methods we use on our UNIX machines. Ideally, we would learn some new approaches with NT that we could then apply to our UNIX systems to help some rising problems on those systems.

In this paper, we describe the software installation methods that we use on our UNIX machines, because they provide a context for our desired result. We explain what we have learned about NT applications and why we feel they should be considered differently than typical UNIX applications. Finally, we describe the system that we have begun to use, explain how it helps us, and identify remaining problems.

This is an interesting time for NT administrators because the range of tools is changing so quickly, due primarily to Microsoft's growing effort to help solve scalable administration problems. Many of the tools that we are now using didn't exist when we started this project, and most likely, some tools will be released between the time this paper was written and the time it will be presented. However, we believe that our desired solution, our observations on applications, and our current approach will still be of use to other administrators who have to surmount similar problems.

2. UNIX Software Installation

When we started our NT application installation project, we initially looked at our existing infrastructure to identify the components that we felt worked well for us. These helped shape our concept of an ideal NT solution. We present a brief overview of our UNIX software installation here.

All of our UNIX machines are interconnected over our internal LAN. We manage several different flavors of UNIX, but we have the same software installation methods for each of them, so we will only discuss the Solaris implementation in detail. Although we don't use Depot [1] because it doesn't solve all of our problems, any readers familiar with that software installation method will recognize the concepts presented here.

The software installed on our Solaris machines can be broken into two different categories:

1. Software that is installed on the local machine. This includes the operating system, some daemons, and any licensed third party software that has to run on a specific system.
2. Software that is installed on a networked file system. This includes the vast majority of software installed on Solaris, including programs that have graphical user interfaces, application suites, and simple command line utilities.

Locally installed software is usually installed at build time by our Solaris build scripts. If we decide to install a new piece of software locally onto a machine, then we install it on every Solaris machine. This is very important - it is much easier to manage all of our Solaris computers as a single unit if each of them has an identical set of installed software. We modify the build scripts to ensure that new Solaris machines also get this software. As mentioned above, this is a rare occurrence.

Most of the installed software on our Solaris machines is installed into a networked file system. In our case, the path to this networked file system is /software/solaris, but this same concept is more commonly available as /usr/local at other sites. Each Solaris computer mounts this single network repository. This means that, once a piece of software is installed in /software/solaris, every Solaris box immediately has access to it. In the overwhelming majority of cases, no extra work is required on any computer in order to make the software available.

The /software/solaris filesystem is exported via NFS. It doesn't deliver software quite as quickly as local disk would, but that has never been an issue. It will scale reasonably to several hundred client machines. Beyond that point, we will replicate the directory and serve it from multiple servers, which will require a minimal amount of work.

One serious problem with this approach is that if the network server goes down, then all of the Solaris hosts are crippled. In our environment of relatively reliable networks and computers and a relatively small number of administrators, we deem this to be an acceptable tradeoff. Further, we replicate the /software/solaris file system to a disk on a second server, and have the clients automatically fail over to the second disk when possible. This helps when the server has a problem, but doesn't solve the case of a network outage. In practice, this has not been an issue. (If the network is down, there are usually bigger problems.)

When a piece of software is installed, we put it in its own directory under /software/solaris. The path is hardcoded to include the software's category and version number. For example, emacs is in /software/solaris/apps/packages/emacs-19.34.6/. All emacs files of this version are stored under this directory. This allows us to install many different versions of emacs without any collisions between them. This is crucial.

Not every package can be easily convinced to live under one directory, but in practice, we have always been able to get around this. UNIX software is typically easy to modify with configurable files, and in the most cases, can be fooled with symbolic links.

All that the user needs to do to access most applications is to put this directory in their PATH environment variable: /software/solaris/apps/bin. We populate this directory with symbolic links to the actual executables that live in their own directories, and, using the "soft" [2] system, we modify the user's startup shells to update any environment variables necessary. We can change the default

version of an application with one quick command. Again, we trade off a minor performance hit (symlink resolution) for increased flexibility of administration.

Our UNIX installation is not without problems. As the size and complexity of applications grows larger, and the cost of local disk grows cheaper, we find that we would like to install more applications on individual machines. Doing so is more problematic than installing them centrally. Also, an increasing number of third-party applications have arbitrary restrictions that nearly force us to do by-hand installations on individual machines. Our general feeling is that in these cases, those vendors are borrowing the worst ideas from the world of PC applications.

The key points of our UNIX installation methods are:

- Each machine is identical to every other machine of the same type.
- Most software is installed in a network file system, and in this case, no extra work is required to make software available on all machines.
- All information related to an application version is kept in one place -- the directory for that software.
- No extra work is required to make software available to all users.
- We make heavy use of symbolic links to glue different areas of the file system together into a seamless interface for users.

3. NT Software Installation

One of the hardest tasks that we have with our NT environment is installing software. Some people have expressed surprise at this. After all, they point out, it's much easier to install an application under NT. All you have to do is put the CD in the drive, click setup, and answer some questions as InstallShield goes to work. One would expect that

UNIX software management would be much harder, since there is no uniform way of installing applications, configuring them requires editing various arcane files, and sometimes they even have to be compiled. This is true; if one's goal is to build a single machine for a single user and install a given set of applications, it is much easier to do on NT than on UNIX.

However, managing a large environment for many different users is entirely different from building an isolated machine. Ironically, it is the NT application's focus on the user and on the single machine that primarily makes it so much harder to maintain on a large scale. But this is not the only difference between software on NT and software on UNIX, and not the only reason that large-scale NT software installation is difficult.

In this section, we identify the issues that have made NT application installation challenging in our environment. It is our hope that some of these issues will be addressed in future releases of NT and the various applications. Understanding these issues (and why we consider them to be issues) is also key to understanding how we install applications.

3.1. NT Software is GUI-based

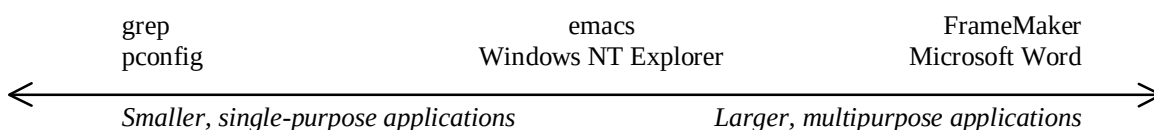
The first difference noted by most people is that NT's applications generally have graphical user interfaces, while UNIX applications may or may not. We find that whether or not an application as a GUI really has no bearing on how difficult it is to install.

A somewhat related issue is whether or not administration tools should be based around GUI. Generally, GUI-based tools are harder to automate and extend. Large environments all have their own particular requirements, so administration tools for them require customization. At the present time, command line tools are easier for us to use when managing a large, diverse environment.

3.2. NT Software is More Complex

Most of the applications that we install on NT have a different focus than those we install on UNIX. To illustrate this point, see the line in Figure One. On the far left-hand side of the line are small, single-

Figure One: Software Continuum



purpose command line utilities such as “cat”, “grep”, and “traceroute”. On the far right are large, multipurpose monolithic applications such as “Office 97” and “Internet Explorer”. These applications tend to provide a very large set of features from a single interface. Applications like emacs and perl lie somewhere in between.

The majority of the applications that we install on UNIX are on the left side of the line, while the majority of the applications that we install on Windows NT are on the right side of the line. This is not to say that UNIX has no large, monolithic productivity-based applications. It certainly does. But in our environment, UNIX owns the left side of the line, while Windows NT owns the right side. We believe that the wealth of these types of applications for NT is the primary reason for its steady rise in popularity.

The focus of the program generally does not have an impact on installation. However, the applications on the left side of the line are often easier to install, regardless of their OS. There are several reasons for this.

- Footprint. The smaller the application, the smaller the fixed storage footprint. When applications are stored on network file servers, the application load time becomes an important factor. For small programs this is often not an issue, but for very large ones, the startup delay can be painful. As a result, one leans toward installing the application on a local disk rather than on a network file server, making it much more difficult to install and update.
- Options. Typically, the larger and more complicated an application, the more options it has at install time. And, during the lifetime of the application, the larger, more comprehensive applications seem to want more add-ons and more components. This makes it quite a bit more difficult to get a single installation that will satisfy a large group of users.
- OS integration. Large applications are very nearly operating systems in and of themselves. (Emacs used to be accused of this, but it has been far outclassed by applications in the NT world.) They depend on a large portion of the OS, and they often have a very strong reliance on the OS being set up in an exact manner. On NT, this has resulted in the applications being tightly integrated with the OS. The problems with this under are described in the next section.

It is interesting to note that we are seeing an increasing number of large, GUI-focused

applications available for UNIX. This is one of the areas where we hope that our lessons learned from NT application installation will help us out on the UNIX side.

Also, there are more utility-style programs available for NT, including ports of perl, various UNIX utilities, and a batch of new, NT-specific tools. Unfortunately, it is still difficult to install networked versions of these tools to be shared by all the workstations from a single location, usually because they are too tightly bound to a specific machine.

3.3. NT Software Lives in a Single-User World

Historically, NT applications come from an environment where they could make certain assumptions that are no longer valid:

- The application would only be installed on a single computer.
- Only one user used the computer.
- The application would be installed on local disk.
- The application might have to add things to the operating system.

This set of assumptions, combined with the usual large size and complexity of NT applications, has resulted in a set of conventions for NT application configuration that make large-scale installation particularly difficult.

- NT applications are too intertwined with the OS. Most applications depend on mfc42.dll or shell32.dll to be present on the machine. When these files aren't there or are different from the expected version, software will replace them with the version that they were built with. This can cause other applications to crash or act oddly. Since they also often copy in other DLLs, they tend to bloat system32, making it more difficult to install network versions of the application, and also rather difficult to decide what is safe to uninstall. Also, applications that are written for Windows95 will often try to write to areas that are protected under NT and require Administrator privileges.
- NT applications are too machine-specific. Most applications write their configuration information into the registry. Since the registry is machine-specific, the software is only configured correctly for that one machine. The usual solution, which doesn't scale well, is to copy portions of the registry to other machines. Discovering which portions of the registry to

copy, and doing so without causing side effects, is a challenge. Some software also makes use of a user's environment variables, which, unlike UNIX, are set up on a per machine basis, as part of the local registry.

- NT applications are user-specific. This shows up in two ways. First, when an application is installed, it will often add itself to the current user's Start menu. Other users, if they login, will not be able to run the application, even though it's installed on the machine. This can be fixed by making sure every application is added to the 'All Users' portion of the Start menu. Not all applications are smart enough to do this yet. Second, applications tend to store user-specific data in various places on the local machine, including the registry, user directories, application directories, and even operating system directories. So, for example, one user may login to a particular machine and be able to read some other user's email because the email program wasn't programmed to handle multiple users.
- Roaming profiles aren't quite right. Roaming profiles attempt to solve some of the above problems, but they create new issues. When applications are installed on a single machine, roaming profiles end up pointing to applications that aren't there. If a roaming profile is loaded onto a workstation that has a differing installation of the same app, they munge the correct settings.

Because of these differences, we have to approach NT software support completely differently than UNIX software support. Overcoming the above problems requires special tools, special care, and a lot of planning.

4. Possible Installation Approaches

In order to decide on which installation methods we'd ultimately use, we had to evaluate and consider as many options as we could. Some of these have been used by other sites, some of these we came up with on our own. Our final solution was a mix of these.

4.1. Installation By Hand

The first option is the obvious: by hand. You sit at the machine, pop in the CD-ROM, answer all the questions, wait while it copies files, you answer some more questions, and eventually you're finished. For a minor optimization of this technique, you can

put copies of the CDs or floppies on the network, and install from that.

The by-hand method is usually done by sitting at the console of the machine, but it's also possible to bring up a remote console of the machine using various third party software, including the SMS Help Desk application. Neither of these is what you really want to do, as the by-hand approach almost always involves too much of time overhead. Worse, for each iteration of the install, you introduce the chance for things to change from machine to machine, such as accidentally selecting different options.

We decided that by-hand installs should be avoided if at all possible. (However, if you have a lot of free labor lying around, this may be the easiest solution.)

4.2. Installation at Build Time

For a default set of software that you know will seldom change, arranging for the software to be automatically installed at build time is ideal. This method is automated, it scales, and it remains consistent for each machine. After an automated build with software install, you have a base machine with a known set of software. You know exactly how that machine is configured.

In order to automatically install software at build time, you usually run a command script after the initial OS install, or initiate an automated administrator login that executes a series of install scripts. In our current solution, we ended up doing both.

The first option, running a command script, we use to apply the latest NT service pack and hot-fixes -- admittedly more of an OS issue than a software issue. We could make more use of this, however the system needs to reboot after the application of the service pack, so we turned to the second option.

To use an automated login, it's best to create an account dedicated to that purpose. That way you can disable it when you're not using it, but, more important to the task, you can assign it a specific login script. The script should execute any post-install cleanup that couldn't be done before the first reboot, and then launch an installation script of your choosing, be it a batch file, a perl script, a CMD file, or what have you. Properly done, this can spawn the appropriate package installs, one at a time, and in the order you specify. The order can be important -- for example, getting perl or Winzip on the system first would be prudent as later installs will want to use them.

We experimented with a third option: using the OEM install tools found in the Resource Kit [3], notably sysdiff. With sysdiff, you take a snapshot of a machine before an installation and a snapshot after. Sysdiff records the differences, which, presumably, can then be applied to any other machine to install that same piece of software. We found, however, that sysdiff did not scale well. We started using it when we were first mass-building our NT environment. As is common in these cases, what we thought should be a default build at the start of the project and what we eventually discovered should have been the default build were vastly different, the latter being a much larger set of applications. With each attempt to add an application to the sysdiff package, it took more and more tweaking to the various configuration files to make the creation of the snapshots actually succeed. Sysdiff also had serious problems when our machines weren't compatible at a hardware level. More often than not, as the default build grew, sysdiff would fail miserably, usually with a fatal error at the last step of the procedure, creating the distribution directory.

However, installing software at build time has one serious problem. It only fixes the installation problem once, which is fine if you never expect to install a new application in the future. However, if you do upgrade your software, you have to modify the software that is installed at boot time. This changes the known base, so you are likely to end up with inconsistent machines. Worse, you probably will also have to install this application on machines that were built months before. This essentially puts you back where you started -- trying to figure out how to install software on existing machines.

Installing software at build time, then, is not a complete solution, but it's certainly useful to be able to do it.

4.3. Automated Remote Installation

What you really want is some way of automatically installing software on remote machines. We considered several different methods.

SMS has other uses besides remote control. It also has a feature called the "package command manager". This allows the administrator to specify software 'packages' that can be installed on workstations in one of two ways. She can schedule compulsory software packages that will be installed at a certain time. For optional software, the user can be presented with a list of software that can be installed at login. The second option helps with the problem of buying licenses that you don't need,

allocating expensive software packages only to those who need them.

In order to use the package command manager, you need to bundle your software into a package. Creating a package involves finding or creating a single-command install option that is usually silent to the user, defaults to all the right options and preferably runs in the background. For some applications, this is trivial. For others, it can be quite painful, but we've yet to find a package we absolutely can't do this with. As a last ditch resort, we wrap the setup program in a Visual Basic program that basically clicks the correct buttons automatically. (This borders on ridiculous, but it works.)

It's also possible to do remote installs without SMS. The administrator wraps the software installation in the same kind of package, and then uses rsh or rlogin to connect to the machine and execute the setup command locally. The Microsoft supplied rsh is less than ideal, so we use the Ataman package. The advantage of using rsh is that it runs when you tell it to, in contrast to Package Command Manager, which has its own internal scheduling mechanisms. If you need to push a hotfix out right away, you will want to use rsh.

An administrator can also use the NT scheduler service to execute commands on remote machines. This is again similar to the package command manager method of installation. We only briefly toyed with this idea, as it involves configuring the machine to run the scheduler service as a user with administrative privileges, and we were unsure of the security aspects involved. The other options seemed to provide the same functionality, but this method is an option to consider.

In essence, remote installation consists of three issues:

- Wrapping the software in some kind of package that essentially turns the entire install into one single command.
- Initiating that command on a remote machine. SMS handles this the most gracefully.
- Worrying about the permissions with which the remote command will execute.

4.4. Installation Via Replication

Another approach is to replicate a disk image onto the local disk. (We've heard of some sites that boot into linux in order to update Windows95 FAT partitions remotely.) The replication can come from a CD or a networked disk. The problem with this

method is that it essentially involves constantly erasing the local disk, which has two side effects.

First, this prevents any user data being stored locally. This may not be so bad if your users are careful about keeping their data on a network server, and if you store profiles on the servers. But we've had interesting problems with that solution, particularly in an environment that has a lot of laptops. In terms of a time investment, this can be almost as bad, or perhaps even worse, than manual software installations.

Second, there's an inherent SID problem replication brings up, in that the machine's SID needs to be retained, or in the case of building new machines, a new SID has to be created. Finally, not all machines in an environment share an identical hardware base.

The replication solution just doesn't scale to a large environment, at least not with today's available replication software.

4.5. Remote Booting

Finally, one can consider remote booting by using a boot PROM on the network card on a machine with no local disk. From an administrative standpoint, this is a dream setup. From a user standpoint, this is the worst possible solution, as you've sacrificed local disk performance for ease of installation. If this is the option you go with, you should probably consider some of the Windows terminals that are available, or wait for the NT 5 caching schemes.

5. Requirements for a Software Installation Solution

When we first started installing NT machines in our workplace, we took the simple solution to installing software: we installed it by hand. That worked fine as long as there were only a few machines, and those of us using them knew exactly what we were doing. But the situation quickly got out of hand, and we realized we were spending a lot of time walking from one machine to the next, a lousy solution that we've never had to implement on our UNIX machines.

At that point, we went through our analysis described above - understanding how NT software is installed, considering various options for large-scale installation, and identifying what we liked about our solution for UNIX.

We decided that whatever solution we developed had to have the following requirements.

- The installed applications need to achieve reasonable performance. Our first attempt was to use a network installation for major software. Performance was miserable. Not only did users complain, but in some cases, the software crashed. This was in part due to slow network storage, and in part because the applications didn't handle non-local disk installation very well. So, this meant that we had to plan to install most software onto local disk.
- The distribution to each machine had to be automated. We refuse to do more walking from machine to machine. We'd like to visit each computer only twice in its lifetime: once to install it, and once to take it away. We should be able to do everything else remotely. Note that this isn't because we're lazy, it's because visiting each computer simply doesn't scale.
- Our solution should attempt to keep the application from interfering with the operating system. This would help reduce mysterious problems with the operating system and collisions with other applications.
- We would like every machine that we manage to be as identical as possible, differing only when the hardware configuring forces it. The operating system, the patch levels, and the applications should otherwise be the same. This allows us to generalize across the machines, to simply know, without looking, that an application is installed or that some command should work. Unfortunately, this doesn't work in the NT world for two reasons. First, no machine has that much local disk space. Second, the licensing cost would overwhelm us.
- More realistically, we decided to define several classes of machines: developer machines, visualization machines, student machines, secretary machines, and so on. Computers in a particular group all have the same software installed on them. This is not quite as good as having one flavor of machine, but it still lets us generalize.
- We've learned not to trust applications that have been known to mess with the OS. We'd like to have a way of discovering what they did at installation, and what they did over time. Ideally we would have a way of checking each machine regularly. This would help us maintain our goal of keeping every machine the same.
- Our solution should be as simple as possible. (It's currently not.) For example, it should be

possible to build a machine and all the software on it by popping a floppy in it, booting, and walking away. Hours later, it should have all software installed on it correctly. Afterwards, when we update software across the net, it should only take one command, and it should automatically happen on all of the appropriate machines.

- We have a lot of students who work with us for short periods of time. They should be able to come up to speed on the tools that we're using relatively quickly. This means that an encyclopedic knowledge of NT and NT tools should not be required.
- Our method needed to help us keep track of installed software so that we could ensure that we are legal with respect to licenses for installed software.
- Finally, we're aware that the NT world changes pretty quickly. We didn't want to spend a lot of time building a custom solution that would be useless in a year. Instead, we wanted to build it out of existing tools, using Microsoft-sanctioned methods, so that we would be more likely to be able to adapt to changes.

6. Our NT Software Installation Approach

There are three main components to our current NT software installation method.

1. A repository of packaged software.
2. A build procedure that installs a pre-determined set of packaged software on new machines.
3. An update process that installs packaged software on existing machines.

When we have a new application to install, we package it, put it in the repository, and arrange for new machines to install it. Then we initiate a network push that installs that software on all existing machines. This solution doesn't quite meet all of our ideals (for example, packaging software is not particularly simple), but it's a rational method that has reduced the amount of legwork and helped to keep our machines running consistent sets of software.

6.1. The Software Repository

The software repository is a networked share where we store software that is ready to be installed. Each of these applications is "packaged", meaning that it

can be completely installed by running a single command.

The majority of the work here comes in creating the package. In many cases, applications already have a silent or unattended install option, and the package simply consists of running 'setup' and directing it to the correct "answer" file for its defaults.

In other cases we have to use additional software to prepare the package for unattended install. Internet Explorer and Microsoft Office, both required the use of their resource kits to create an unattended distribution.

And then there are the applications that have no concept of unattended install. For these, we write simple VB apps which send the correct key sequences to the installer applets.

6.2. The Build Procedure

We use the NT 4.0 unattended installation procedure and our own boot floppy to initiate the install process. We found that we can significantly reduce the initial build time by following the Microsoft recommendation to remove portions of the NT distribution from the distribution location on the network. The initial build time, which includes formatting a portion of the workstation disk, takes about 25 minutes per machine on a fast ethernet network. The initial distribution applies the latest service packs and hot-fixes, installing them by using the "run at install time" feature in unattended install mode.

Once the OS is installed, our build procedure adds a registry key for auto-logon of a predetermined account at boot. The machine reboots, and logs in as that account. The logon script for that account registers the machine with the SMS database. Next, the script initiates the step that installs all of the packaged software. First it looks to see what class of machines this computer is (for example, secretary or developer), from which it can determine which applications need to be installed. This length of time that this takes depends on the number of applications that we add, but 10 to 15 minutes is about average.

Finally, the login script makes some local registry modifications, copies in some shortcuts, and adds printers.

Setting the network correctly is tricky, especially since the machine is usually moved from our lab to someone's desktop after it has been built, which may require that it get a new network address. When the machine is initially built, we use DHCP to let assign an IP address from the server's pool of available

addresses. This is how DHCP servers are normally used. However, we don't use Dynamic DNS, and we want DNS to work for every machine, so we have to make sure that once a machine is built, it gets the same IP address from that point on. We accomplish this by configuring the DHCP server to reserve a specific IP address for a client MAC address. We obtain the MAC address from SMS during the part of the build process in which the new machine adds itself to the SMS database.

6.3. The Update Process

Once a piece of software is packaged, installing it on a remote machine is simply a matter of remotely invoking the package with the right permissions. A number of ways to do that are discussed in Section 4.3.

For the most part, we use SMS to push software packages out, because it runs with the right permissions and it gracefully handles machines that are temporarily off. Occasionally, if we're in a hurry, we use rsh rather than SMS. At the moment, these pushes are initiated by hand, but they could be automated to ensure that every machine of a certain class always has the correct software.

6.4. Observations on Our Approach

The approach we've taken is working very well for us. We integrated the best parts of what we researched, and came up with a reasonably automated system that requires little administrator presence at the machine after the initial build.

Being able to specify that all machines get the latest hot-fix without having to leave our office is a big win from an administrative standpoint. In fact, that very situation came up at our site after the recent denial of service attacks aimed at .gov sites.

We are able to keep a steady base level of machines by ensuring that when a software package is added to the default build for new machines, it also gets installed on the existing base of machines. This keeps guesswork at a minimum when it comes to troubleshooting or license tracking. It's about as close as we can come to a unix environment, where all applications sit in a common nfs-shared repository. We at least can be assured of a common platform.

Using these methods, the only differences between machines within the same group (general workstation, development, etc) are at a hardware and driver level. It's still not the perfect world, but it's at least as level a playing field as unix.

We have a few problems left. We don't have a way of carefully monitoring all machines to see what has changed during an installation, and we occasionally run into problems with this. While we think our current solution is pretty simple, packaging a piece of software can sometimes be fairly tricky. Our biggest problems, though, are associated with the basic NT installation issues that we can't fix: the tendency of an application to be too closely tied to a specific machine and user.

7. Comparing NT and UNIX Installation Methodology

Having implemented fairly comprehensive software installation methods for both NT and UNIX, we find that neither of our solutions is completely satisfactory. Indeed, there are features of our UNIX machines that we wish we could use on NT, and aspects of the NT installs that would be very handy on UNIX.

7.1. Our UNIX Wishlist

Here are the features of our NT application installation that we wish we could easily duplicate on UNIX:

- Better fault tolerance. When the UNIX file server with all the applications crashes, which is rare, everyone is in trouble. On NT machines, nearly all of the applications are local, so people can continue to work even when the network crashes. One way to fix this with UNIX would be to have smarter automounters that can handle server failover. Another would be to install applications locally on each machine, just as we do now with NT.
- A common push/pull method that's better than rdist.
- Even fewer machine-specific applications. We're worried about the trend we see in some UNIX software that forces us to install applications locally and then node-locks the licenses.
- A one-step configure, build, and install mechanism similar to NT's InstallShield.

7.2. Our NT Wishlist

- Symbolic links.
- A clean, enforced separation of the operating system, applications, and user space.
- A clean separation of multiple versions of the same application. It's very handy to be able to

try out a beta version of an application without having to move completely away from the stable version.

- More useful profiles. Profiles go a long way towards capturing a user's configuration, but not enough applications use them, and roaming profiles just don't work right. Storing profiles on the server, with options to copy, link or create machine-specific profiles, similar to the irix method, would be a better solution.
- Less reliance on the local registry. Too many applications use it for storing configuration and state information. This doesn't allow for a consistent work environment for users that use more than one machine. We'd like to see software installs stay away from the HKLM registry tree altogether, and instead of editing HKCU, putting configuration settings into a globally available repository.
- Consistent silent installation mechanisms. There are too many different installer technologies for Windows, each with its own learning curve and methodology, and many of these installers don't support a silent install option, forcing ugly hacks.

8. Conclusions

We set out to develop a scalable software installation procedure for NT that was as useful to us as our existing UNIX installation strategy. After learning about NT application issues, many tools for NT installation, and trying out various options, we have developed a solution that, while unlike UNIX strategy, is still fairly scalable. During the process, we identified some features of NT that we would like to replicate on UNIX, and vice-versa.

Our remaining difficulties with large-scale support of NT application installation are intrinsic to the way in which NT applications interact with the operating system. We know that some of these issues will be addressed in future versions of NT, and look forward to that event.

9. Author and Project Information

Michail Gomberg is a systems administrator in the Mathematics and Computer Science Division at Argonne National Laboratory. He was the lead technical architect for this project. His email address is gomberg@mcs.anl.gov.

Rémy Evard is the manager of Advanced Computing Technologies and Networks in the Mathematics and

Computer Science Division at Argonne National Laboratory. He is actively pursuing research in systems administration, with the hope of making it less difficult and more fun. His email address is evard@mcs.anl.gov.

Craig Stacey is a systems administrator in the Mathematics and Computer Science Division at Argonne National Laboratory. He was the one whose soles were saved with this procedure. His email address is stace@mcs.anl.gov.

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

10. References

- [1] Colyer, Wallace & Wong, Walter, *Depot: A Tool For Managing Software Environments*, LISA VI Conference Proceedings, 1992.
- [2] Evard, Rémy and Leslie, Robert, *Soft: A Software Environment Abstraction Mechanism*, LISA VIII Conference Proceedings, 1994.
- [3] Microsoft Windows NT Workstation 4.0 Resource Kit, Microsoft Corporation, Microsoft Press, 1996.

Appendix A: Tools

The following is a list of many of the tools that we've found to be very useful.

NT Resource Kit:

Srvinfo	A very useful utility to look at services on a remote machine from the command line.
Shortcut	Allows us to copy .lnk files to the right places without remaining linked to the source location.
Windiff	A graphical comparison tool that can be used to look directory differences.
Instsrv	Allows installation of services from the command line.
rkillsv	Allows processes to be killed remotely.
rconsole	Remote console for NT.
setupmgr	GUI used to create a base answer file for the unattended setup. We had to do a lot of tuning to this file to get it to be really useful.

NT Rollout tools:

Sysdiff	Creates snapshots and difference databases of a machine after software is installed.
IEAK	Internet Explorer administrator kit. It creates a custom distribution, which can then be installed in unattended mode.
ORK	Office resource kit. Same function as the IEAK, but for MS Office.
Winnt	This is the NT installer that comes on the NT 4.0 CD. We use it to install NT in unattended mode.
regedit	Merges registry edits into the machine registry.

NT Services:

Dfs	Lets us create a single distribution location, spread over multiple disks and multiple servers.
SMS	SMS provides a database of clients on the network, along with the ability to schedule commands for

execution across the clients. Also includes software auditing and remote control.

SMS Package Command Manager

This service comes with BackOffice resource kit and allows SMS jobs to run unattended.

DHCP Server	We use DHCP to get the machine onto the network initially, without having to assign an IP address at build time. Once the machine is built, we use DHCP to assign a specific IP address to it, by getting the MAC address from SMS.
-------------	---

Scheduler Service

Can be used to run jobs remotely, however there are security issues regarding 'system' account.

Other Microsoft Tools:

DOS 6.22	Boot disks need to be DOS 6.22 in order to fit all required files on it. Windows95 or 98 system files take up too much room.
MSLANMAN	We use LAN Manager for DOS to copy the NT files onto the workstation disk.
MSDN CDs	Incredibly useful for getting the latest information, especially pertaining to the reduced TCO initiatives from Microsoft.

Windows Development Tools:

Visual Basic	We use VB to run queries on the SMS database, and create automated installations for applications that don't have an unattended install option.
--------------	---

Third party:

Ataman rsh	Execute commands remotely on machines.
Perl5	We use the Activeware version. Without this we would be sunk, or at least very unhappy.

Exhibit 0002A

conference

proceedings

**Large Installation
System Administration of
Windows NT Conference
Proceedings**

*Seattle, Washington
August 5-8, 1998*

Math
TK5105
.5
.L375
1998

ed by The **USENIX** Association and
System Administrators Guild

USENIX

The Advanced Computing
Systems Association

SAGE
THE SYSTEM ADMINISTRATORS GUILD

USENIX Association

**Proceedings of the
Large Installation System Administration of Windows NT
Conference**

**August 5-8, 1998
Seattle, Washington**

Table of Contents

Large Installation System Administration of Windows NT Conference

August 5-8, 1998
Seattle, Washington

Thursday, August 6

Management and Monitoring

Session Chair: Chris Rouland, Internet Security Systems, Inc.

Patch 32: A System for Automated Client OS Updates 1
Gerald Carter, Auburn University

Monitoring Utilization in an NT Workstation Lab 11
Paul Kranenburg, Erasmus University Rotterdam

Jointly Managing UNIX and NT

Session Chair: Mark Verber, WebTV

File System Security: Secure Network Data Sharing for NT and Unix 17
Bridget Allison, Rob Hawley, Andrea Borr, Mark Muhlestein, and Dave Hitz, Network Appliance

NT Domains for UNIX 85
Luke Kenneth Casson Leighton

Friday, August 7

Operating System and Software Installation

Session Chair: Richard Holland, Rockwell Collins, Inc.

AutoInstall for NT: Complete NT Installation Over the Network 27
Robert Fulmer and Alex Levine, Lucent Technologies, Bell Labs

A Comparison of Large-Scale Software Installation Methods on NT and UNIX 37
Michail Gomberg, Rémy Evard, and Craig Stacey, Argonne National Laboratory

Software Distribution to PC Clients in an Enterprise Network 49
Cameron D. Luerkens, H. John Cole, and Danielle R. Legg, Rockwell Collins, Inc.

Saturday, August 8

Architecture and Servers

Session Chair: Michael Carpenter, Collective Technologies

Compaq's New Engineering Compute Farm Environment: Moving Forward with Windows NT 55
Don Brace, Andrew Gordon, and Scott Teel, Compaq Computer Corporation

Designing an Optimized Enterprise Windows NT/95/98 Client Backup Solution With Intelligent
Data Management 69
Kevin M. Workman, Earl Waud, Steven Downs, and Mikel Featherstone, Qualcomm Inc.

Providing Reliable NT Desktop Services by Avoiding NT Server 75
*Thomas A. Limoncelli, Robert Fulmer, Thomas Reingold, Alex Levine, and Ralph Loura,
Lucent Technologies, Bell Labs*

A Comparison of Large-Scale Software Installation Methods on NT and UNIX

Michail Gomberg, Rémy Evard, & Craig Stacey

Mathematics and Computer Science Division, Argonne National Laboratory

Abstract

Our computing environment consists of hundreds of UNIX and NT-based computers. We have a coherent UNIX software installation model that scales comfortably to hundreds of machines. We have spent a great deal of time in the last year learning to understand the software installation and support mechanisms for a similarly large NT infrastructure. In this paper, we examine the underlying requirements for large-scale software installation support, and compare and contrast the NT and UNIX environments, identifying strong points, weak points, and issues.

1. Introduction

The computing environment in the Mathematics and Computer Science Division of Argonne National Laboratory consists of nearly a thousand computers, including supercomputers, servers, workstations, desktop machines, and laptops. The majority of these are running UNIX, but the number of NT machines is steadily growing, particularly on desktops.

This set of machines is managed by a group of seven system administrators, meaning that, like most systems administration groups, we have more than enough work to go around. Our ability to administer the environment relies on scalable techniques. We have to manage large sets of machine as if they were one single system, most preferably from one single location. We were originally nervous about our ability to do this in a growing Windows NT environment due to its reputation as a system requiring hands-on administration. Software installation was one such area that we were worried about.

On each of our UNIX architectures, including SunOS, Solaris, Linux, FreeBSD, AIX, and IRIX installations, we use a common method of installing software. It scales well, in that we are able to install a given piece of software once and have it automatically work on every machine of a given type. This system is described in detail below.

As we began to look at software installation on NT in detail, we began to feel that our fears were justified. The usual software installation methods on NT require the administrator to sit in front of an individual machine, answer questions interactively, wait for the software to load, possibly reboot the machine, and then do a lot of tweaking in order to make the software available to other people who might log in to the machine. This approach doesn't scale to hundreds of machines. Fortunately, there are ways around most of these problems, but, as described below, they are not uniformly applicable to all NT applications. As a result, we started a long-term project to understand NT application installation better, to test the various installation tools, and to develop a philosophy and method of software installation that scaled as well as the methods we use on our UNIX machines. Ideally, we would learn some new approaches with NT that we could then apply to our UNIX systems to help some rising problems on those systems.

In this paper, we describe the software installation methods that we use on our UNIX machines, because they provide a context for our desired result. We explain what we have learned about NT applications and why we feel they should be considered differently than typical UNIX applications. Finally, we describe the system that we have begun to use, explain how it helps us, and identify remaining problems.

This is an interesting time for NT administrators because the range of tools is changing so quickly, due primarily to Microsoft's growing effort to help solve scalable administration problems. Many of the tools that we are now using didn't exist when we started this project, and most likely, some tools will be released between the time this paper was written and the time it will be presented. However, we believe that our desired solution, our observations on applications, and our current approach will still be of use to other administrators who have to surmount similar problems.

2. UNIX Software Installation

When we started our NT application installation project, we initially looked at our existing infrastructure to identify the components that we felt worked well for us. These helped shape our concept of an ideal NT solution. We present a brief overview of our UNIX software installation here.

All of our UNIX machines are interconnected over our internal LAN. We manage several different flavors of UNIX, but we have the same software installation methods for each of them, so we will only discuss the Solaris implementation in detail. Although we don't use Depot [1] because it doesn't solve all of our problems, any readers familiar with that software installation method will recognize the concepts presented here.

The software installed on our Solaris machines can be broken into two different categories:

1. Software that is installed on the local machine. This includes the operating system, some daemons, and any licensed third party software that has to run on a specific system.
2. Software that is installed on a networked file system. This includes the vast majority of software installed on Solaris, including programs that have graphical user interfaces, application suites, and simple command line utilities.

Locally installed software is usually installed at build time by our Solaris build scripts. If we decide to install a new piece of software locally onto a machine, then we install it on every Solaris machine. This is very important - it is much easier to manage all of our Solaris computers as a single unit if each of them has an identical set of installed software. We modify the build scripts to ensure that new Solaris machines also get this software. As mentioned above, this is a rare occurrence.

Most of the installed software on our Solaris machines is installed into a networked file system. In our case, the path to this networked file system is `/software/solaris`, but this same concept is more commonly available as `/usr/local` at other sites. Each Solaris computer mounts this single network repository. This means that, once a piece of software is installed in `/software/solaris`, every Solaris box immediately has access to it. In the overwhelming majority of cases, no extra work is required on any computer in order to make the software available.

The `/software/solaris` filesystem is exported via NFS. It doesn't deliver software quite as quickly as local disk would, but that has never been an issue. It will scale reasonably to several hundred client machines. Beyond that point, we will replicate the directory and serve it from multiple servers, which will require a minimal amount of work.

One serious problem with this approach is that if the network server goes down, then all of the Solaris hosts are crippled. In our environment of relatively reliable networks and computers and a relatively small number of administrators, we deem this to be an acceptable tradeoff. Further, we replicate the `/software/solaris` file system to a disk on a second server, and have the clients automatically fail over to the second disk when possible. This helps when the server has a problem, but doesn't solve the case of a network outage. In practice, this has not been an issue. (If the network is down, there are usually bigger problems.)

When a piece of software is installed, we put it in its own directory under `/software/solaris`. The path is hardcoded to include the software's category and version number. For example, emacs is in `/software/solaris/apps/packages/emacs-19.34.6/`. All emacs files of this version are stored under this directory. This allows us to install many different versions of emacs without any collisions between them. This is crucial.

Not every package can be easily convinced to live under one directory, but in practice, we have always been able to get around this. UNIX software is typically easy to modify with configurable files, and in the most cases, can be fooled with symbolic links.

All that the user needs to do to access most applications is to put this directory in their PATH environment variable: `/software/solaris/apps/bin`. We populate this directory with symbolic links to the actual executables that live in their own directories, and, using the "soft" [2] system, we modify the user's startup shells to update any environment variables necessary. We can change the default

version of an application with one quick command. Again, we trade off a minor performance hit (symlink resolution) for increased flexibility of administration.

Our UNIX installation is not without problems. As the size and complexity of applications grows larger, and the cost of local disk grows cheaper, we find that we would like to install more applications on individual machines. Doing so is more problematic than installing them centrally. Also, an increasing number of third-party applications have arbitrary restrictions that nearly force us to do by-hand installations on individual machines. Our general feeling is that in these cases, those vendors are borrowing the worst ideas from the world of PC applications.

The key points of our UNIX installation methods are:

- Each machine is identical to every other machine of the same type.
- Most software is installed in a network file system, and in this case, no extra work is required to make software available on all machines.
- All information related to an application version is kept in one place -- the directory for that software.
- No extra work is required to make software available to all users.
- We make heavy use of symbolic links to glue different areas of the file system together into a seamless interface for users.

3. NT Software Installation

One of the hardest tasks that we have with our NT environment is installing software. Some people have expressed surprise at this. After all, they point out, it's much easier to install an application under NT. All you have to do is put the CD in the drive, click setup, and answer some questions as InstallShield goes to work. One would expect that

UNIX software management would be much harder, since there is no uniform way of installing applications, configuring them requires editing various arcane files, and sometimes they even have to be compiled. This is true; if one's goal is to build a single machine for a single user and install a given set of applications, it is much easier to do on NT than on UNIX.

However, managing a large environment for many different users is entirely different from building an isolated machine. Ironically, it is the NT application's focus on the user and on the single machine that primarily makes it so much harder to maintain on a large scale. But this is not the only difference between software on NT and software on UNIX, and not the only reason that large-scale NT software installation is difficult.

In this section, we identify the issues that have made NT application installation challenging in our environment. It is our hope that some of these issues will be addressed in future releases of NT and the various applications. Understanding these issues (and why we consider them to be issues) is also key to understanding how we install applications.

3.1. NT Software is GUI-based

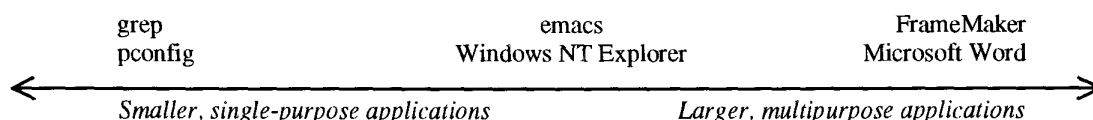
The first difference noted by most people is that NT's applications generally have graphical user interfaces, while UNIX applications may or may not. We find that whether or not an application as a GUI really has no bearing on how difficult it is to install.

A somewhat related issue is whether or not administration tools should be based around GUI. Generally, GUI-based tools are harder to automate and extend. Large environments all have their own particular requirements, so administration tools for them require customization. At the present time, command line tools are easier for us to use when managing a large, diverse environment.

3.2. NT Software is More Complex

Most of the applications that we install on NT have a different focus than those we install on UNIX. To illustrate this point, see the line in Figure One. On the far left-hand side of the line are small, single-

Figure One: Software Continuum



purpose command line utilities such as “cat”, “grep”, and “tracert”. On the far right are large, multipurpose monolithic applications such as “Office 97” and “Internet Explorer”. These applications tend to provide a very large set of features from a single interface. Applications like emacs and perl lie somewhere in between.

The majority of the applications that we install on UNIX are on the left side of the line, while the majority of the applications that we install on Windows NT are on the right side of the line. This is not to say that UNIX has no large, monolithic productivity-based applications. It certainly does. But in our environment, UNIX owns the left side of the line, while Windows NT owns the right side. We believe that the wealth of these types of applications for NT is the primary reason for its steady rise in popularity.

The focus of the program generally does not have an impact on installation. However, the applications on the left side of the line are often easier to install, regardless of their OS. There are several reasons for this.

- Footprint. The smaller the application, the smaller the fixed storage footprint. When applications are stored on network file servers, the application load time becomes an important factor. For small programs this is often not an issue, but for very large ones, the startup delay can be painful. As a result, one leans toward installing the application on a local disk rather than on a network file server, making it much more difficult to install and update.
- Options. Typically, the larger and more complicated an application, the more options it has at install time. And, during the lifetime of the application, the larger, more comprehensive applications seem to want more add-ons and more components. This makes it quite a bit more difficult to get a single installation that will satisfy a large group of users.
- OS integration. Large applications are very nearly operating systems in and of themselves. (Emacs used to be accused of this, but it has been far outclassed by applications in the NT world.) They depend on a large portion of the OS, and they often have a very strong reliance on the OS being set up in an exact manner. On NT, this has resulted in the applications being tightly integrated with the OS. The problems with this under are described in the next section.

It is interesting to note that we are seeing an increasing number of large, GUI-focused

applications available for UNIX. This is one of the areas where we hope that our lessons learned from NT application installation will help us out on the UNIX side.

Also, there are more utility-style programs available for NT, including ports of perl, various UNIX utilities, and a batch of new, NT-specific tools. Unfortunately, it is still difficult to install networked versions of these tools to be shared by all the workstations from a single location, usually because they are too tightly bound to a specific machine.

3.3. NT Software Lives in a Single-User World

Historically, NT applications come from an environment where they could make certain assumptions that are no longer valid:

- The application would only be installed on a single computer.
- Only one user used the computer.
- The application would be installed on local disk.
- The application might have to add things to the operating system.

This set of assumptions, combined with the usual large size and complexity of NT applications, has resulted in a set of conventions for NT application configuration that make large-scale installation particularly difficult.

- NT applications are too intertwined with the OS. Most applications depend on mfc42.dll or shell32.dll to be present on the machine. When these files aren't there or are different from the expected version, software will replace them with the version that they were built with. This can cause other applications to crash or act oddly. Since they also often copy in other DLLs, they tend to bloat system32, making it more difficult to install network versions of the application, and also rather difficult to decide what is safe to uninstall. Also, applications that are written for Windows95 will often try to write to areas that are protected under NT and require Administrator privileges.
- NT applications are too machine-specific. Most applications write their configuration information into the registry. Since the registry is machine-specific, the software is only configured correctly for that one machine. The usual solution, which doesn't scale well, is to copy portions of the registry to other machines. Discovering which portions of the registry to

copy, and doing so without causing side effects, is a challenge. Some software also makes use of a user's environment variables, which, unlike UNIX, are set up on a per machine basis, as part of the local registry.

- NT applications are user-specific. This shows up in two ways. First, when an application is installed, it will often add itself to the current user's Start menu. Other users, if they login, will not be able to run the application, even though it's installed on the machine. This can be fixed by making sure every application is added to the 'All Users' portion of the Start menu. Not all applications are smart enough to do this yet. Second, applications tend to store user-specific data in various places on the local machine, including the registry, user directories, application directories, and even operating system directories. So, for example, one user may login to a particular machine and be able to read some other user's email because the email program wasn't programmed to handle multiple users.
- Roaming profiles aren't quite right. Roaming profiles attempt to solve some of the above problems, but they create new issues. When applications are installed on a single machine, roaming profiles end up pointing to applications that aren't there. If a roaming profile is loaded onto a workstation that has a differing installation of the same app, they munge the correct settings.

Because of these differences, we have to approach NT software support completely differently than UNIX software support. Overcoming the above problems requires special tools, special care, and a lot of planning.

4. Possible Installation Approaches

In order to decide on which installation methods we'd ultimately use, we had to evaluate and consider as many options as we could. Some of these have been used by other sites, some of these we came up with on our own. Our final solution was a mix of these.

4.1. Installation By Hand

The first option is the obvious: by hand. You sit at the machine, pop in the CD-ROM, answer all the questions, wait while it copies files, you answer some more questions, and eventually you're finished. For a minor optimization of this technique, you can

put copies of the CDs or floppies on the network, and install from that.

The by-hand method is usually done by sitting at the console of the machine, but it's also possible to bring up a remote console of the machine using various third party software, including the SMS Help Desk application. Neither of these is what you really want to do, as the by-hand approach almost always involves too much of time overhead. Worse, for each iteration of the install, you introduce the chance for things to change from machine to machine, such as accidentally selecting different options.

We decided that by-hand installs should be avoided if at all possible. (However, if you have a lot of free labor lying around, this may be the easiest solution.)

4.2. Installation at Build Time

For a default set of software that you know will seldom change, arranging for the software to be automatically installed at build time is ideal. This method is automated, it scales, and it remains consistent for each machine. After an automated build with software install, you have a base machine with a known set of software. You know exactly how that machine is configured.

In order to automatically install software at build time, you usually run a command script after the initial OS install, or initiate an automated administrator login that executes a series of install scripts. In our current solution, we ended up doing both.

The first option, running a command script, we use to apply the latest NT service pack and hot-fixes -- admittedly more of an OS issue than a software issue. We could make more use of this, however the system needs to reboot after the application of the service pack, so we turned to the second option.

To use an automated login, it's best to create an account dedicated to that purpose. That way you can disable it when you're not using it, but, more important to the task, you can assign it a specific login script. The script should execute any post-install cleanup that couldn't be done before the first reboot, and then launch an installation script of your choosing, be it a batch file, a perl script, a CMD file, or what have you. Properly done, this can spawn the appropriate package installs, one at a time, and in the order you specify. The order can be important -- for example, getting perl or Winzip on the system first would be prudent as later installs will want to use them.

We experimented with a third option: using the OEM install tools found in the Resource Kit [3], notably sysdiff. With sysdiff, you take a snapshot of a machine before an installation and a snapshot after. Sysdiff records the differences, which, presumably, can then be applied to any other machine to install that same piece of software. We found, however, that sysdiff did not scale well. We started using it when we were first mass-building our NT environment. As is common in these cases, what we thought should be a default build at the start of the project and what we eventually discovered should have been the default build were vastly different, the latter being a much larger set of applications. With each attempt to add an application to the sysdiff package, it took more and more tweaking to the various configuration files to make the creation of the snapshots actually succeed. Sysdiff also had serious problems when our machines weren't compatible at a hardware level. More often than not, as the default build grew, sysdiff would fail miserably, usually with a fatal error at the last step of the procedure, creating the distribution directory.

However, installing software at build time has one serious problem. It only fixes the installation problem once, which is fine if you never expect to install a new application in the future. However, if you do upgrade your software, you have to modify the software that is installed at boot time. This changes the known base, so you are likely to end up with inconsistent machines. Worse, you probably will also have to install this application on machines that were built months before. This essentially puts you back where you started -- trying to figure out how to install software on existing machines.

Installing software at build time, then, is not a complete solution, but it's certainly useful to be able to do it.

4.3. Automated Remote Installation

What you really want is some way of automatically installing software on remote machines. We considered several different methods.

SMS has other uses besides remote control. It also has a feature called the "package command manager". This allows the administrator to specify software 'packages' that can be installed on workstations in one of two ways. She can schedule compulsory software packages that will be installed at a certain time. For optional software, the user can be presented with a list of software that can be installed at login. The second option helps with the problem of buying licenses that you don't need,

allocating expensive software packages only to those who need them.

In order to use the package command manager, you need to bundle your software into a package. Creating a package involves finding or creating a single-command install option that is usually silent to the user, defaults to all the right options and preferably runs in the background. For some applications, this is trivial. For others, it can be quite painful, but we've yet to find a package we absolutely can't do this with. As a last ditch resort, we wrap the setup program in a Visual Basic program that basically clicks the correct buttons automatically. (This borders on ridiculous, but it works.)

It's also possible to do remote installs without SMS. The administrator wraps the software installation in the same kind of package, and then uses rsh or rlogin to connect to the machine and execute the setup command locally. The Microsoft supplied rsh is less than ideal, so we use the Ataman package. The advantage of using rsh is that it runs when you tell it to, in contrast to Package Command Manager, which has its own internal scheduling mechanisms. If you need to push a hotfix out right away, you will want to use rsh.

An administrator can also use the NT scheduler service to execute commands on remote machines. This is again similar to the package command manager method of installation. We only briefly toyed with this idea, as it involves configuring the machine to run the scheduler service as a user with administrative privileges, and we were unsure of the security aspects involved. The other options seemed to provide the same functionality, but this method is an option to consider.

In essence, remote installation consists of three issues:

- Wrapping the software in some kind of package that essentially turns the entire install into one single command.
- Initiating that command on a remote machine. SMS handles this the most gracefully.
- Worrying about the permissions with which the remote command will execute.

4.4. Installation Via Replication

Another approach is to replicate a disk image onto the local disk. (We've heard of some sites that boot into linux in order to update Windows95 FAT partitions remotely.) The replication can come from a CD or a networked disk. The problem with this

method is that it essentially involves constantly erasing the local disk, which has two side effects.

First, this prevents any user data being stored locally. This may not be so bad if your users are careful about keeping their data on a network server, and if you store profiles on the servers. But we've had interesting problems with that solution, particularly in an environment that has a lot of laptops. In terms of a time investment, this can be almost as bad, or perhaps even worse, than manual software installations.

Second, there's an inherent SID problem replication brings up, in that the machine's SID needs to be retained, or in the case of building new machines, a new SID has to be created. Finally, not all machines in an environment share an identical hardware base.

The replication solution just doesn't scale to a large environment, at least not with today's available replication software.

4.5. Remote Booting

Finally, one can consider remote booting by using a boot PROM on the network card on a machine with no local disk. From an administrative standpoint, this is a dream setup. From a user standpoint, this is the worst possible solution, as you've sacrificed local disk performance for ease of installation. If this is the option you go with, you should probably consider some of the Windows terminals that are available, or wait for the NT 5 caching schemes.

5. Requirements for a Software Installation Solution

When we first started installing NT machines in our workplace, we took the simple solution to installing software: we installed it by hand. That worked fine as long as there were only a few machines, and those of us using them knew exactly what we were doing. But the situation quickly got out of hand, and we realized we were spending a lot of time walking from one machine to the next, a lousy solution that we've never had to implement on our UNIX machines.

At that point, we went through our analysis described above - understanding how NT software is installed, considering various options for large-scale installation, and identifying what we liked about our solution for UNIX.

We decided that whatever solution we developed had to have the following requirements.

- The installed applications need to achieve reasonable performance. Our first attempt was to use a network installation for major software. Performance was miserable. Not only did users complain, but in some cases, the software crashed. This was in part due to slow network storage, and in part because the applications didn't handle non-local disk installation very well. So, this meant that we had to plan to install most software onto local disk.
- The distribution to each machine had to be automated. We refuse to do more walking from machine to machine. We'd like to visit each computer only twice in its lifetime: once to install it, and once to take it away. We should be able to do everything else remotely. Note that this isn't because we're lazy, it's because visiting each computer simply doesn't scale.
- Our solution should attempt to keep the application from interfering with the operating system. This would help reduce mysterious problems with the operating system and collisions with other applications.
- We would like every machine that we manage to be as identical as possible, differing only when the hardware configuring forces it. The operating system, the patch levels, and the applications should otherwise be the same. This allows us to generalize across the machines, to simply know, without looking, that an application is installed or that some command should work. Unfortunately, this doesn't work in the NT world for two reasons. First, no machine has that much local disk space. Second, the licensing cost would overwhelm us.
- More realistically, we decided to define several classes of machines: developer machines, visualization machines, student machines, secretary machines, and so on. Computers in a particular group all have the same software installed on them. This is not quite as good as having one flavor of machine, but it still lets us generalize.
- We've learned not to trust applications that have been known to mess with the OS. We'd like to have a way of discovering what they did at installation, and what they did over time. Ideally we would have a way of checking each machine regularly. This would help us maintain our goal of keeping every machine the same.
- Our solution should be as simple as possible. (It's currently not.) For example, it should be

possible to build a machine and all the software on it by popping a floppy in it, booting, and walking away. Hours later, it should have all software installed on it correctly. Afterwards, when we update software across the net, it should only take one command, and it should automatically happen on all of the appropriate machines.

- We have a lot of students who work with us for short periods of time. They should be able to come up to speed on the tools that we're using relatively quickly. This means that an encyclopedic knowledge of NT and NT tools should not be required.
- Our method needed to help us keep track of installed software so that we could ensure that we are legal with respect to licenses for installed software.
- Finally, we're aware that the NT world changes pretty quickly. We didn't want to spend a lot of time building a custom solution that would be useless in a year. Instead, we wanted to build it out of existing tools, using Microsoft-sanctioned methods, so that we would be more likely to be able to adapt to changes.

6. Our NT Software Installation Approach

There are three main components to our current NT software installation method.

1. A repository of packaged software.
2. A build procedure that installs a pre-determined set of packaged software on new machines.
3. An update process that installs packaged software on existing machines.

When we have a new application to install, we package it, put it in the repository, and arrange for new machines to install it. Then we initiate a network push that installs that software on all existing machines. This solution doesn't quite meet all of our ideals (for example, packaging software is not particularly simple), but it's a rational method that has reduced the amount of legwork and helped to keep our machines running consistent sets of software.

6.1. The Software Repository

The software repository is a networked share where we store software that is ready to be installed. Each of these applications is "packaged", meaning that it

can be completely installed by running a single command.

The majority of the work here comes in creating the package. In many cases, applications already have a silent or unattended install option, and the package simply consists of running 'setup' and directing it to the correct "answer" file for its defaults.

In other cases we have to use additional software to prepare the package for unattended install. Internet Explorer and Microsoft Office, both required the use of their resource kits to create an unattended distribution.

And then there are the applications that have no concept of unattended install. For these, we write simple VB apps which send the correct key sequences to the installer applets.

6.2. The Build Procedure

We use the NT 4.0 unattended installation procedure and our own boot floppy to initiate the install process. We found that we can significantly reduce the initial build time by following the Microsoft recommendation to remove portions of the NT distribution from the distribution location on the network. The initial build time, which includes formatting a portion of the workstation disk, takes about 25 minutes per machine on a fast ethernet network. The initial distribution applies the latest service packs and hot-fixes, installing them by using the "run at install time" feature in unattended install mode.

Once the OS is installed, our build procedure adds a registry key for auto-logon of a predetermined account at boot. The machine reboots, and logs in as that account. The logon script for that account registers the machine with the SMS database. Next, the script initiates the step that installs all of the packaged software. First it looks to see what class of machines this computer is (for example, secretary or developer), from which it can determine which applications need to be installed. This length of time that this takes depends on the number of applications that we add, but 10 to 15 minutes is about average.

Finally, the login script makes some local registry modifications, copies in some shortcuts, and adds printers.

Setting the network correctly is tricky, especially since the machine is usually moved from our lab to someone's desktop after it has been built, which may require that it get a new network address. When the machine is initially built, we use DHCP to let assign an IP address from the server's pool of available

addresses. This is how DHCP servers are normally used. However, we don't use Dynamic DNS, and we want DNS to work for every machine, so we have to make sure that once a machine is built, it gets the same IP address from that point on. We accomplish this by configuring the DHCP server to reserve a specific IP address for a client MAC address. We obtain the MAC address from SMS during the part of the build process in which the new machine adds itself to the SMS database.

6.3. The Update Process

Once a piece of software is packaged, installing it on a remote machine is simply a matter of remotely invoking the package with the right permissions. A number of ways to do that are discussed in Section 4.3.

For the most part, we use SMS to push software packages out, because it runs with the right permissions and it gracefully handles machines that are temporarily off. Occasionally, if we're in a hurry, we use rsh rather than SMS. At the moment, these pushes are initiated by hand, but they could be automated to ensure that every machine of a certain class always has the correct software.

6.4. Observations on Our Approach

The approach we've taken is working very well for us. We integrated the best parts of what we researched, and came up with a reasonably automated system that requires little administrator presence at the machine after the initial build.

Being able to specify that all machines get the latest hot-fix without having to leave our office is a big win from an administrative standpoint. In fact, that very situation came up at our site after the recent denial of service attacks aimed at .gov sites.

We are able to keep a steady base level of machines by ensuring that when a software package is added to the default build for new machines, it also gets installed on the existing base of machines. This keeps guesswork at a minimum when it comes to troubleshooting or license tracking. It's about as close as we can come to a unix environment, where all applications sit in a common nfs-shared repository. We at least can be assured of a common platform.

Using these methods, the only differences between machines within the same group (general workstation, development, etc) are at a hardware and driver level. It's still not the perfect world, but it's at least as level a playing field as unix.

We have a few problems left. We don't have a way of carefully monitoring all machines to see what has changed during an installation, and we occasionally run into problems with this. While we think our current solution is pretty simple, packaging a piece of software can sometimes be fairly tricky. Our biggest problems, though, are associated with the basic NT installation issues that we can't fix: the tendency of an application to be too closely tied to a specific machine and user.

7. Comparing NT and UNIX Installation Methodology

Having implemented fairly comprehensive software installation methods for both NT and UNIX, we find that neither of our solutions is completely satisfactory. Indeed, there are features of our UNIX machines that we wish we could use on NT, and aspects of the NT installs that would be very handy on UNIX.

7.1. Our UNIX Wishlist

Here are the features of our NT application installation that we wish we could easily duplicate on UNIX:

- Better fault tolerance. When the UNIX file server with all the applications crashes, which is rare, everyone is in trouble. On NT machines, nearly all of the applications are local, so people can continue to work even when the network crashes. One way to fix this with UNIX would be to have smarter automounters that can handle server failover. Another would be to install applications locally on each machine, just as we do now with NT.
- A common push/pull method that's better than rdist.
- Even fewer machine-specific applications. We're worried about the trend we see in some UNIX software that forces us to install applications locally and then node-locks the licenses.
- A one-step configure, build, and install mechanism similar to NT's InstallShield.

7.2. Our NT Wishlist

- Symbolic links.
- A clean, enforced separation of the operating system, applications, and user space.
- A clean separation of multiple versions of the same application. It's very handy to be able to

try out a beta version of an application without having to move completely away from the stable version.

- More useful profiles. Profiles go a long way towards capturing a user's configuration, but not enough applications use them, and roaming profiles just don't work right. Storing profiles on the server, with options to copy, link or create machine-specific profiles, similar to the irix method, would be a better solution.
- Less reliance on the local registry. Too many applications use it for storing configuration and state information. This doesn't allow for a consistent work environment for users that use more than one machine. We'd like to see software installs stay away from the HKLM registry tree altogether, and instead of editing HKCU, putting configuration settings into a globally available repository.
- Consistent silent installation mechanisms. There are too many different installer technologies for Windows, each with its own learning curve and methodology, and many of these installers don't support a silent install option, forcing ugly hacks.

8. Conclusions

We set out to develop a scalable software installation procedure for NT that was as useful to us as our existing UNIX installation strategy. After learning about NT application issues, many tools for NT installation, and trying out various options, we have developed a solution that, while unlike UNIX strategy, is still fairly scalable. During the process, we identified some features of NT that we would like to replicate on UNIX, and vice-versa.

Our remaining difficulties with large-scale support of NT application installation are intrinsic to the way in which NT applications interact with the operating system. We know that some of these issues will be addressed in future versions of NT, and look forward to that event.

9. Author and Project Information

Michail Gomberg is a systems administrator in the Mathematics and Computer Science Division at Argonne National Laboratory. He was the lead technical architect for this project. His email address is gomberg@mcs.anl.gov.

Rémy Evard is the manager of Advanced Computing Technologies and Networks in the Mathematics and

Computer Science Division at Argonne National Laboratory. He is actively pursuing research in systems administration, with the hope of making it less difficult and more fun. His email address is evard@mcs.anl.gov.

Craig Stacey is a systems administrator in the Mathematics and Computer Science Division at Argonne National Laboratory. He was the one whose soles were saved with this procedure. His email address is stace@mcs.anl.gov.

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

10. References

- [1] Colyer, Wallace & Wong, Walter, *Depot: A Tool For Managing Software Environments*, LISA VI Conference Proceedings, 1992.
- [2] Evard, Rémy and Leslie, Robert, *Soft: A Software Environment Abstraction Mechanism*, LISA VIII Conference Proceedings, 1994.
- [3] Microsoft Windows NT Workstation 4.0 Resource Kit, Microsoft Corporation, Microsoft Press, 1996.

Appendix A: Tools

The following is a list of many of the tools that we've found to be very useful.

NT Resource Kit:

Srvinfo	A very useful utility to look at services on a remote machine from the command line.
Shortcut	Allows us to copy .lnk files to the right places without remaining linked to the source location.
Windiff	A graphical comparison tool that can be used to look directory differences.
Instsrv	Allows installation of services from the command line.
rkillsv	Allows processes to be killed remotely.
rconsole	Remote console for NT.
setupmgr	GUI used to create a base answer file for the unattended setup. We had to do a lot of tuning to this file to get it to be really useful.

NT Rollout tools:

Sysdiff	Creates snapshots and difference databases of a machine after software is installed.
IEAK	Internet Explorer administrator kit. It creates a custom distribution, which can then be installed in unattended mode.
ORK	Office resource kit. Same function as the IEAK, but for MS Office.
Winnt	This is the NT installer that comes on the NT 4.0 CD. We use it to install NT in unattended mode.
regedit	Merges registry edits into the machine registry.

NT Services:

Dfs	Lets us create a single distribution location, spread over multiple disks and multiple servers.
SMS	SMS provides a database of clients on the network, along with the ability to schedule commands for

execution across the clients. Also includes software auditing and remote control.

SMS Package Command Manager

This service comes with BackOffice resource kit and allows SMS jobs to run unattended.

DHCP Server	We use DHCP to get the machine onto the network initially, without having to assign an IP address at build time. Once the machine is built, we use DHCP to assign a specific IP address to it, by getting the MAC address from SMS.
-------------	---

Scheduler Service

Can be used to run jobs remotely, however there are security issues regarding 'system' account.

Other Microsoft Tools:

DOS 6.22	Boot disks need to be DOS 6.22 in order to fit all required files on it. Windows95 or 98 system files take up too much room.
MSLANMAN	We use LAN Manager for DOS to copy the NT files onto the workstation disk.
MSDN CDs	Incredibly useful for getting the latest information, especially pertaining to the reduced TCO initiatives from Microsoft.

Windows Development Tools:

Visual Basic	We use VB to run queries on the SMS database, and create automated installations for applications that don't have an unattended install option.
--------------	---

Third party:

Ataman rsh	Execute commands remotely on machines.
Perl5	We use the Activeware version. Without this we would be sunk, or at least very unhappy.

PENN STATE UNIVERSITY LIBRARIES



A000032962666

ISBN 1-880446-96-0

Exhibit 0002B



- [Bookmarks 0](#)
- [Course Reserves](#)
- [My Account](#)
- [Advanced search](#)

Search in

Search Bar

Keyword

▼

search for

Search...

Search

- [Advanced search](#)
- [Start Over](#)

Actions for Proceedings of the Large Installation System Administration of Windows NT Conference : August 5-8, 1998, Seattle, Washington

[Share](#)
[Email](#) [RIS file](#)
[Bookmark](#)
[Report an Issue](#)

Proceedings of the Large Installation System Administration of Windows NT Conference : August 5-8, 1998, Seattle, Washington / USENIX Association

Conference Author
[Large Installation System Administration of Windows NT Conference \(1998 : Seattle, Wash.\)](#)
Published
Berkeley, CA : USENIX Association, [1998]
Copyright Date
©1998
Physical Description
88 pages : illustrations ; 28 cm
Additional Creators
[USENIX Association](#) and [System Administrators Guild](#)

Availability

[Browse Nearby on Shelf](#)
[I Want It](#)
[I Want It](#)

Physical & Mathematical Sciences Library (UP) (1 item)

Listing where to find this item in the library.

Call number	Material	Location
-------------	----------	----------

Call number	Material	Location
TK5105.5.L375 1998 Book	Physical & Mathematical Sciences Library, 201 Davey Lab - General Collection	

Subject(s)

- [Microsoft Windows NT—Congresses](#)
- [UNIX \(Computer file\)—Congresses](#)
- [Computer networks—Management—Congresses](#)
- [Large scale systems—Congresses](#)

ISBN

1880446960

Note

"Co-sponsored by the USENIX Association and SAGE, the System Administrators Guild"--Cover.

Bibliography Note

Includes bibliographical references.

[View MARC record](#) | catkey: 1712083

Ask
a Librarian



PennState

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a [Creative Commons Attribution 4.0 license](#). [Details and exceptions](#).

[Legal Statements](#) | [PSU Hotlines](#)

Penn State University Libraries

- [Libraries Home](#)
- [Libraries Intranet \(Staff Only\)](#)
- [Accessibility Help](#)
- [Website Feedback](#)
- [Policies and Guidelines](#)
- [Staff Directory](#)
- [\(814\) 865-6368](#)

Connect with Penn State University Libraries

- [Facebook](#)
- [Twitter](#)
- [Instagram](#)



Keyword



Search...

Search

[Advanced search](#) [Start Over](#)

Share

Bookmark

Report an Issue

MARC View

LEADER 01430nam a22003491a 4500

001 1712083

003 SIRSI

005 20151215094614.0

008 980910t19981998xx 000 0 eng d

019 a | MARS

020 a | 1880446960

035 a | LIAS2124262

035 a | (OCoLC)39953370

040 a | PSt c | PSt d | WaOLN d | UtOrBLW

050 4 a | TK5105.5 b | .L375 1998

111 2 a | Large Installation System Administration of Windows NT Conference d | (1998 : c | Seattle, Wash.)

245 1 0 a | Proceedings of the Large Installation System Administration of Windows NT Conference : b | August 5-8, 1998, Seattle, Washington :
USENIX Association.

264 1 a | Berkeley, CA : b | USENIX Association, c | [1998]

264 4 c | ©1998

300 a | 88 pages : b | illustrations ; c | 28 cm

336 a | text b | txt 2 | rdacontent

337 a | unmediated b | n 2 | rdamedia

338 a | volume b | nc 2 | rdacarrier



338	a	volume b nc 2 rdacarrier
500	a	"Co-sponsored by the USENIX Association and SAGE, the System Administrators Guild"--Cover.
504	a	Includes bibliographical references.
630	0	0 a Microsoft Windows NT v Congresses.
630	0	0 a UNIX (Computer file) v Congresses.
650	0	a Computer networks x Management v Congresses.
650	0	a Large scale systems v Congresses.
710	2	a USENIX Association.
710	2	a System Administrators Guild.
949	a	TK5105.5.L375 1998 w LC c 1 i 000032962666 d 9/10/1998 e 10/26/1998 l STACKS-PM m UP-PAMS n 3 r Y s Y t BOOK u 4/13/2001

ASK
A LIBRARIAN



PennState

Copyright ©2019 The Pennsylvania State University. All rights reserved. Except where otherwise noted, this work is subject to a Creative Commons Attribution 4.0 license. Details and exceptions.

[Legal Statements](#) | [PSU Hotlines](#)

PENN STATE UNIVERSITY LIBRARIES

[Libraries Home](#) (814) 865-6368
[Libraries Intranet \(Staff Only\)](#)
[Accessibility Help](#)
[Website Feedback](#)
[Policies and Guidelines](#)
[Staff Directory](#)

CONNECT WITH PENN STATE UNIVERSITY LIBRARIES

[Facebook](#)
[Twitter](#)
[Instagram](#)

Exhibit 0002C

← → ↻ <https://web.archive.org/web/19980209162244/http://www.lias.psu.edu/> 120% ☆

<http://www.lias.psu.edu/> Go OCT FEB MAY 1997 09 1998 About this cap

284 captures
18 Oct 1996 – 29 Mar

LIAS Penn State
Libraries Information Access System

View and Renew
Books You've
Borrowed


[As of Feb. 02](#)

 **The CAT**
Books & more

 [FAST TRACK to All Resources](#)

 **The VEL**
Web Virtual Library
PSU & Big Ten

 [ARTICLES in Journals/Newspapers](#)

 [TELL ME WHERE to Look](#)

[Construction
Update](#)

[The CAT](#)

[Search](#)

[Libraries Home](#)

[LIAS Seminars](#)

[HELP](#)

[Comments](#)

[Libraries
Hours](#)

[Library Computing Services](#) develops LIAS, and works with the University
Libraries to provide electronic resources and services to Penn State.

Exhibit 0002D



LIBRARIES
SEARCH

JOURNAL
SEARCH

BROWSE

Sign in

Menu ▼



BOOK

☆

**Proceedings of the Large Installation System
Administration of Windows NT Conference :
August 5-8, 1998, Seattle, Washington**

Large Installation System Administration of Windows NT Conference
(1998 : Seattle, Wash.); USENIX Association; System Administrators
Guild
Berkeley, CA : USENIX Association; ©1998

📖 Available at TC Walter Library Sci/Eng Books (Level F) (Quarto
TK5105.5 .L375x 1998) >

Get It

Please sign in to see request options and to place requests. Sign in

< BACK TO LOCATIONS

LOCATION ITEMS

TC Walter Library
Available , Sci/Eng Books (Level F) ; Quarto TK5105.5 .L375x 1998
(1 copy, 1 available, 0 requests)



Item in place	31951P00910233N	▼
Loanable		

Report an issue

Cite or Share

24/7 chat



CITATION



E-MAIL



GET LINK



EASYBIB



ENDNOTE
BASIC



EXPORT BIBTEX



EXPORT TO
EXCEL



EXPORT RIS



PRINT



QR

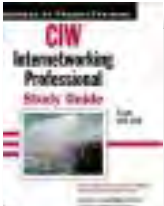
https://primo.lib.umn.edu/permalink/01UMN_INST/1msd8mg/alma9942259960001701

 COPY LINK TO CLIPBOARD

Details

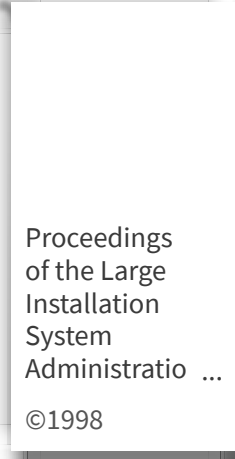
Title	Proceedings of the Large Installation System Administration of Windows NT Conference : August 5-8, 1998, Seattle, Washington
Responsibility	USENIX Association.
Author/Creator	Large Installation System Administration of Windows NT Conference (1998 : Seattle, Wash.) > USENIX Association > System Administrators Guild >
Publisher	Berkeley, CA : USENIX Association
Date	©1998
Copyright date	©1998.
Format	88 pages : illustrations ; 28 cm
Language	English
Description	"Co-sponsored by the USENIX Association and SAGE, the System Administrators Guild"--Cover. Includes bibliographical references.
Subjects	Microsoft Windows NT -- Congresses > UNIX (Computer file) -- Congresses > Computer networks -- Management > Large scale systems -- Congresses >
Genre	Congresses >
Identifier	ISBN: 1880446960 OCLC : 39953370
Source	University of Minnesota Catalog 9942259960001701

Related Titles

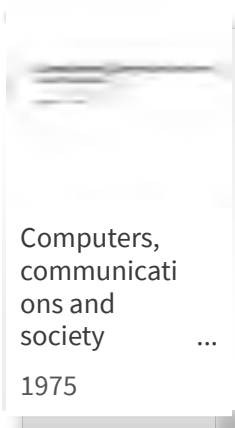


<

networki
essional :
y guide ...
!



Proceedings
of the Large
Installation
System
Administratio ...
©1998



Computers,
communications and
society ...
1975



Network
management
: a practical
perspective ...
©1996



Inte
thin
arch
arch
IoT :
2018

>

https://primo.lib.umn.edu/discovery/fulldisplay?context=L&vid=01UM...pe=TwinCitiesCampus&tab=LibraryCatalog&docid=alma9942259960001701

Page 3 of 4

Nintendo Exhibit 1007

Page 00238

Links

[This item at other libraries \(WorldCat\) !\[\]\(3e9d26d26984df9b84a9484ac9c2d02b_img.jpg\) >](#)

[Display Source Record !\[\]\(ec4cd101c8ae3dc265b7b6791acdb884_img.jpg\) >](#)

[Privacy Statement | Acceptable Use Policy](#)

```

leader 01608nam a2200433ua 4500
001 9942259960001701
005 20240224221259.0
008 980910s1998 caua b 000 0 eng d
020 ##$a1880446960
035 ##$a(0CoLC)39953370
040 ##$aUPM $cUPM $dMnU
049 ##$aMNUA
050 #4$aTK5105.5 $b.L375 1998
111 2#$aLarge Installation System Administration of Windows NT Conference $d(1998 : $cSeattle, Wash.)
245 10$aProceedings of the Large Installation System Administration of Windows NT Conference : $bAugust 5-8, 1998, Seattle,
260 ##$aBerkeley, CA : $bUSENIX Association, $c©1998.
300 ##$a88 pages : $billustrations ; $c28 cm
336 ##$atext $btxt $2rdacontent
337 ##$aunmediated $bn $2rdamedia
338 ##$avolume $bnc $2rdacarrier
500 ##$a"Co-sponsored by the USENIX Association and SAGE, the System Administrators Guild"--Cover.
504 ##$aIncludes bibliographical references.
630 00$aMicrosoft Windows NT $xCongresses.
630 00$aUNIX (Computer file) $xCongresses.
650 #0$aComputer networks $xManagement $vCongresses
650 #0$aLarge scale systems $xCongresses.
710 2#$aUSENIX Association
710 2#$aSystem Administrators Guild
926 ##$aMARS
935 ##$a(MnU)notisARZ4060
935 ##$a(Aleph)000581924UMN01
945 ##$aCAT $b1998-10-14 $nkat1
967 ##$ascs20170325
967 ##$ascs20220128
968 ##$aRDA ENRICHED
999 99$aARZ4060 $bUL
998 ##$aLDR:01610nam#a2200433ua#4500
998 ##$b008:980910s1998###caua####b####000#0#eng#d
997 ##$a39953370

```

[Privacy Statement](#) | [Acceptable Use Policy](#)

Exhibit 0002E

libraries.org

A directory of libraries throughout the world

- Home »
- Libraries »
- Guides
- Documents
- Vendors »
- Products »
- News
- Procurement
- Member Login »

found 15 items where the item contains University of Minnesota. Showing item 7 of 15.

<< prev

list

next >>

search:

University of Minnesota Libraries

University of Minnesota -- Twin Cities

Address: 309 19th Avenue South
Minneapolis, Minnesota
55455
United States



Connect to: [Library Web Site](#) ✓ [Online Catalog](#)

Library details: University of Minnesota Libraries is an Academic library.
This library is affiliated with [University of Minnesota \(view map\)](#) . The collection of the library contains 6,918,302 volumes. The library circulates 610,116 items per year. The library serves a population of 43,943 FTE Enrollment.

Other Info: Member of the Big Ten Academic Alliance

Permalink: <https://librarytechnology.org/library/1024>
(Use this link to refer back to this listing.)

Administration: The director of the library is Wendy Pradt Lougee .

Organizational structure: The library is part of a publicly-funded organization. It operates on a Non-profit financial model.

Academic Level: The library Serves an educational institution at the University level. The Carnegie Foundation designates this institution as: Doctoral Universities: Very High Research Activity (level 15).



Affiliations: The library is a member of the [Association of Research Libraries](#)

Statistics Academic		
Enrollment	43,943	FTE Students
Collection size	6,918,302	volumes
Annual Circulation	610,116	transactions



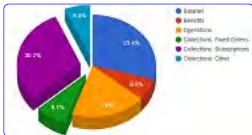
O. Meredith Wilson Library



map location ✓

Related Libraries

- Other libraries with Carnegie class: Doctoral Universities: Very High Research Activity
- All members of the Association of Research Libraries members
- Directory of ARL Member Libraries
- Libraries located in Minneapolis (Minnesota)
- Libraries located in Hennepin county (Minnesota)
- View map of libraries in Hennepin County
- all Academic libs in Minnesota



[View Detailed Statistics](#)

Technology Profile

	Product Name	Year Contracted
Current Automation System	Alma	2013
Previous Automation System	ALEPH 500	2002
Previous Automation System	NOTIS	1986
Discovery Service (w/index)	Ex Libris CDI	
Discovery Interface	Primo	
Authentication Service	EZproxy	
Reading List Manager	Leganto	2018
Interlibrary Loan or Resource Sharing	Resource Sharing for Groups	
Resource Sharing Networks	Big Ten Academic Alliance	
Web Content Management	Drupal	
The library's automation system is hosted by the vendor through an Software-as-a-Service (SaaS) arrangement.		
This library is responsible for the procurement of the library automation system.		

[View Privacy and Security Report](#)

Identifiers

libraries.org ID	1024
NCES LIBID	174066

Record History: This listing was created on Jun 12, 1997 and was last modified on Apr 10, 2025.

Updates: Corrections or Updates? Registered members of Library Technology Guides can submit updates to library listings in libraries.org. [Registration](#) is free and easy. Already registered? [login](#). Or, you can report corrections just by sending a message to [Marshall Breeding](#).

[libraries.org](#)

[Update this Entry](#)

[Library Technology Guides](#)

Maintained by [Marshall Breeding](#)

[terms of service](#)

Copyright 1994-2025

About libraries.org

libraries.org is a free directory of libraries throughout the world.

This directory is a component of [Library Technology Guides](#).

Please [contact Marshall Breeding](#) to report corrections about information listed for this library.

Exhibit 0002F



Conference Proceedings

USENIX Conference Proceedings (by Year and Location)

Abstracts to papers are available to everyone. The full papers are available only to USENIX members. When accessing papers, remember to use your *member number* as your *userid*. To become a USENIX member, please see our [Membership Information](#).

1998:

[Twelfth Systems Administration Conference \(LISA '98\), Boston, Massachusetts](#)

[Sixth Tcl/Tk Conference](#)

[3rd USENIX Workshop on Electronic Commerce](#)

[Large Installation System Administration of Windows NT Conference](#)

[2nd USENIX Windows NT Symposium](#)

[USENIX 1998 Annual Technical Conference, New Orleans, Louisiana](#)

The Wayback Machine - <https://web.archive.org/web/19990116232824/http://www.usenix.org:80/publications/library/proc...>



Conference Proceedings

USENIX Conference Proceedings (by Year and Location)

Abstracts to papers are available to everyone. The full papers are available only to USENIX members. When accessing papers, remember to use your *member number* as your *userid*. To become a USENIX member, please see our [Membership Information](#).

1998:

[Twelfth Systems Administration Conference \(LISA '98\), Boston, Massachusetts](#)

[Sixth Tcl/Tk Conference](#)

[3rd USENIX Workshop on Electronic Commerce](#)

[Large Installation System Administration of Windows NT Conference](#)

[2nd USENIX Windows NT Symposium](#)

[USENIX 1998 Annual Technical Conference, New Orleans, Louisiana](#)

[4th Conference on Object-Oriented Technologies and Systems, Santa Fe, New Mexico](#)

[7th USENIX Security Symposium, San Antonio, Texas](#)

1997:

[USENIX Symposium on Internet Technologies and Systems, Monterey, California](#)

[Eleventh Systems Administration Conference \(LISA '97\), San Diego, California](#)

[Conference on Domain-Specific Languages, Santa Barbara, California](#)

[Large-Scale System Administration of Windows NT Workshop, Seattle, Washington](#)

[USENIX Windows NT Workshop, Seattle, Washington](#)

[Fifth Tcl/Tk Workshop '97, Boston, Massachusetts](#)

[Third USENIX Conference on Object-Oriented Technologies and Systems \(COOTS\), Portland, Oregon](#)

[USENIX 1997 Annual Technical Conference, Anaheim, California](#)

1996:

[The Second USENIX Workshop on Electronic Commerce, Oakland, California](#)

[The Second Symposium on Operating Systems Design and Implementation \(OSDI '96\), Seattle, Washington](#)

[The Tenth Systems Administration Conference \(LISA '96\), Chicago, Illinois](#)

[The Sixth USENIX Security Symposium, San Jose, California](#)

[The Fourth Annual Tcl/Tk Workshop '96, Monterey, California](#)

[The Second USENIX Conference on Object-Oriented Technologies \(COOTS\), Toronto, Ontario, Canada](#)

[USENIX 1996 Annual Technical Conference, San Diego, California](#)

1995:

[USENIX 1995 Technical Conference on UNIX and Advanced Computing Systems, New Orleans, Louisiana,](#)

[Symposium on Mobile and Location-Independent Computing, Ann Arbor, Michigan](#)

[The Fifth USENIX UNIX Security Symposium, Salt Lake City, Utah](#)

[USENIX Conference on Object-Oriented Technologies \(COOTS\), Monterey, California](#)

[Tcl/Tk Workshop, Toronto, Canada](#)

[The First USENIX Workshop on Electronic Commerce, New York, New York](#)

[The Ninth Systems Administration Conference LISA '95, Monterey, California](#)

1994:

[Winter Technical Conference, San Francisco](#)

[C++ Conference, Cambridge, Massachusetts](#)

[UNIX Applications Development Symposium, Toronto, Canada](#)

[Summer Technical Conference, Boston, Massachusetts](#)

[High Speed Networking Symposium, Berkeley, California](#)

[Systems Administration Conference \(LISA VIII\) San Diego, California](#)

[Very High Level Languages Symposium \(VHLL\), Santa Fe, New Mexico](#)

[Operating Systems Design and Implementation Symposium \(OSDI\), Monterey, California](#)

1993:

[Winter Technical Conference, San Diego, California](#)

[Mach III Symposium, Santa Fe, New Mexico](#)

[Summer Technical Conference, Cincinnati, Ohio](#)

[Mobile Computing and Location Independent Symposium,
Cambridge, Massachusetts](#)

[Microkernel Symposium, San Diego, California](#)

[SEDMS IV, San Diego, California](#)

[Security IV, Santa Clara, California](#)

[Systems Administration Conference \(LISA VII\), Monterey,
California](#)

Note: This listing will be updated as we convert our archives.

[How to Order](#)

 Need help? [Use our Contacts page.](#)

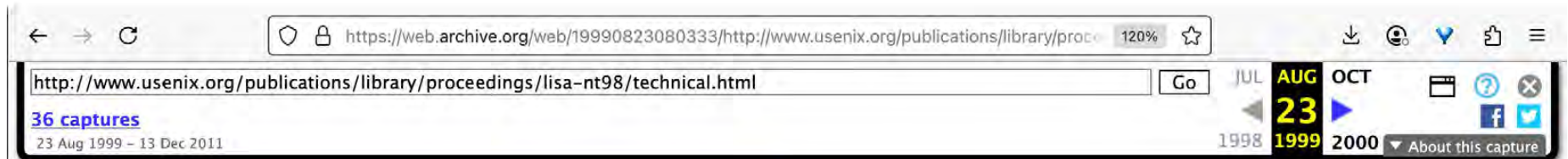
Last changed: 13 Jan 1999 jr

 [Library index](#)

 [Publications index](#)

 [USENIX home](#)

Exhibit 0002G



USENIX Technical Program

**Large Installation System Administration of
Windows NT Conference, 1998**
August 5-8, 1998
Seattle, Washington, USA

Thursday, August 6

Management and Monitoring
Session Chair: Chris Rouland, *Lehman Brothers*

[Patch 32: A System for Automated Client OS Updates](#)
Gerald Carter, *Auburn University*

[Monitoring Utilization in an NT Workstation Lab](#)
Paul Kranenburg, *Erasmus University Rotterdam*

The Wayback Machine - <https://web.archive.org/web/19990823080333/http://usenix.org:80/publications/library/proceedin...>



**Large Installation System Administration of
Windows NT Conference, 1998
August 5-8, 1998
Seattle, Washington, USA**

Thursday, August 6

Management and Monitoring
Session Chair: Chris Rouland, *Lehman Brothers*

[Patch 32: A System for Automated Client OS Updates](#)

Gerald Carter, *Auburn University*

[Monitoring Utilization in an NT Workstation Lab](#)

Paul Kranenburg, *Erasmus University Rotterdam*

Jointly Managing UNIX and NT
Session Chair: Mark Verber, *WebTV*

[File System Security: Secure Network Data Sharing for NT and Unix](#)

Bridget Allison, Rob Hawley, Andrea Borr, Mark Muhlestein,
and Dave Hitz, *Network Appliance*
[Presentation slides](#)

[NT 3.5 / 4.0 Domains for UNIX](#)

Luke Kenneth Casson Leighton

Friday, August 7

Operating System and Software Installation
Session Chair: Richard Holland, *Rockwell Collins, Inc.*

[AutoInstall for NT: Complete NT Installation Over the Network](#)

Robert Fulmer and Alex Levine, *Lucent Technologies, Bell Labs*

[A Comparison of Large-Scale Software Installation Methods on NT and UNIX](#)

Michail Gomborg, Rémy Evard, and Craig Stacey, *Argonne National Laboratory*

[Software Distribution to PC Clients in an Enterprise Network](#)

Cameron D. Luerkens, H. John Cole, and Danielle R. Legg,
Rockwell Collins, Inc.

Saturday, August 8

Architecture and Servers

Session Chair: Michael Carpenter, *Collective Technologies*

**[Compaq's New Engineering Compute Farm Environment:
Moving Forward with Windows NT](#)**

Don Brace, Andrew Gordon, and Scott Teel, *Compaq Computer Corporation*

**[Designing an Optimized Enterprise Windows NT/95/98
Client Backup Solution With Intelligent Data Management](#)**

Kevin Workman, Earl Waud, Steven Downs, and Mikel Featherstone, *Qualcomm Inc.*

[Providing Reliable NT Desktop Services by Avoiding NT Server](#)

Thomas A Limoncelli, Robert Fulmer, Thomas Reingold, Alex Levine, and Ralph Loura, *Lucent Technologies, Bell Labs*

webster@usenix.org

Last changed: 14 Aug 1998 efc

 [Conference Index](#)

 [Proceedings Index](#)

 [USENIX home](#)

Exhibit 0002H

← → ↻ https://web.archive.org/web/19991113081156/http://www.usenix.org/publications/library/pr 120%

http://www.usenix.org/publications/library/proceedings/lisa-nt98/gomberg.html OCT NOV JAN
1998 13 1999 2001 About this capture

48 captures
13 Nov 1999 - 9 Jul 2024

USENIX Abstracts

A Comparison of Large-Scale Software Installation Methods on NT and UNIX

Michail Gomberg, Rémy Evard, and Craig Stacey
Argonne National Laboratory

Abstract

Our computing environment consists of hundreds of UNIX and NT-based computers. We have a coherent UNIX software installation model that scales comfortably to hundreds of machines. We have spent a great deal of time in the last year learning to understand the software installation and support mechanisms for a similarly large NT infrastructure. In this paper, we examine the underlying requirements for large-scale software installation support, and compare and contrast the NT and UNIX environments, identifying strong points, weak points, and issues.

- USENIX Members may view the full text of this paper in [HTML](#) form and [PDF](#) form. (Use your membership number as the username for access.)

The Wayback Machine - <https://web.archive.org/web/19991113081156/http://www.usenix.org:80/publications/library/proc...>



A Comparison of Large-Scale Software Installation Methods on NT and UNIX

Michail Gomberg, Rémy Evard, and Craig Stacey
Argonne National Laboratory

Abstract

Our computing environment consists of hundreds of UNIX and NT-based computers. We have a coherent UNIX software installation model that scales comfortably to hundreds of machines. We have spent a great deal of time in the last year learning to understand the software installation and support mechanisms for a similarly large NT infrastructure. In this paper, we examine the underlying requirements for large-scale software installation support, and compare and contrast the NT and UNIX environments, identifying strong points, weak points, and issues.

- USENIX Members may view the full text of this paper in [HTML](#) form and [PDF](#) form. (Use your membership number as the username for access.)
- If you need the latest Adobe Acrobat Reader, you can download it from [Adobe's site](#).
- To become a USENIX Member, please see our [Membership Information](#).
- Current USENIX Members may change their [password](#).

webster@usenix.org

Last changed: July 24, 1998 prowillen

-  [Technical Program](#)
-  [Conference Index](#)
-  [USENIX home](#)

Exhibit 0002I

Ex. 0002I
Citations to Gomberg

Source Document:

Gomberg, M., Evard, R., & Stacey, C. (1998, August). A comparison of large-scale software installation methods on NT and UNIX. In *Proceedings of the Large Installation System Administration of Windows NT Conference 1998*. **Ex. 0002**

Citations: 2000 – 2001 (in chronological order)

Anderson, P. (2000, April). System Administration: Large-Scale Linux Configuration Management. *Linux Journal*, 2000, 19.

Augusto, A., Guimarães, C., & de Geus, P. L. (2000). Administration Techniques for Implementing Security on Large Windows NT Networks. *Proceedings of SSI*, 20.

Augusto, A., Guimarães, C. C., & de Geus, P. L. (2001, March). DoIt4Me: a tool for automating administrative tasks on Windows NT networks. In *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg)* (pp. 32-37). SBC.

Augusto, A. (2001, September). *Automatização de Administração e Segurança em Redes Windows NT* (Doctoral dissertation, [sn]). Instituto de Computação Universidade Estadual de Campinas (English translation by Google Translate: Computer Institute State University of Campinas).

Augusto, A., Guimarães, C., & de Geus, P. L. (2001). Administration of Large Windows NT Network with DoIt4Me. In *The 10th International Conference on System Administration, Networking and Security*.

Augusto, A. (2001). Applying Security Configurations to a Large Number of Windows NT Computers Without Visiting Each Machine. In *LANOMS* (pp. 75-86).