TECHNICAL CHAPTERS

THE ONLY WAY IT COULD WORK

TUNING THE SYSTEM

THE TRADEMARKS

TUMBLING THROUGH THE DOCUVERSE

THE PROTOCOLS

LITERARY 4/1 MACHINES

THE ONLY WAY IT COULD WORK

THE ONLY WAY IT CAN POSSIBLY BE DONE

Some conventional methods, such as B-trees, permit rapid insertion and deletion in large structures. The slowdown as structure grows is logarithmic.

The ideas promoted in this book could not possibly be contemplated unless methods of storage, editing and linking could be found which all, singly and in combination, deteriorated in performance as a logarithmic function of the size of a document and the size of the docuverse. We believe we have achieved this. As in other dynamic-function problems, analytic proof is not possible, so this is an empirical question to be proven or disproven.

TUNING THE SYSTEM

TUNINGS

The system's design is a unified whole, but we may think of it as a combination of structures: the basic conceptual structure, plus a technical structure which makes it possible, and a contractual structure which makes it possible for people to use it confidently. These aspects taken together make a unified design. Because the conceptual structure required very fast lookup within a tightly organized but large linked system, we had to develop a particular technical structure; and because the conceptual structure expects participants to behave in certain ways, these are embraced in the contract offered to users. These provisions are necessary for the orderly and confident use of published material by many people.

We are concerned with the balance of customer incentives to help foster our overall goals. In the coalescing final design of the system, contracts, categories of service and pricing are all subject to reconsideration. We need to study possible cost functions for reducing possible Babel; or for cutting less-recent accessibilities in order to be practical.

The system has two business commandments, viz.:

1. EVERYBODY MAKES MONEY: there exist many opportunities for profitable participation.

2. ALL SERVICES MUST BE SELF-SUPPORTING.

The following discussions will investigate ramifications of the latter premise.

ROYALTIES FOR WHAT EXACTLY?

Granted that royalties should be exactly proportional to something, what should that be?

If we make it "transmissions," some paradoxes surface. For instance, if you the user have a fancy computer, you and your program may request many transmissions that are used little or not at all-- while certain materials, already transmitted, stay on your screen.

Fairness would suggest that the material on the screen, not what goes over the wire, should be the royalty divisor. This would require certain back-reporting by the front end and may be a can of worms, but it certainly has elements of fairness.

We have considered schemes for getting reports back from the user systems-- optional with the user-- stating, on an honor-system basis, where the royalty-fragments should go that are not measurable at our end as transmissions. But this may be too much flex; and many clowns would award the royalties to their own published works. So there is a case against this just on the basis of straightforwardness.

Or consider the user who has a low rate of transmissions, say 50% of the channel capacity if he chose to ask for fetches at full rate. Should 50% of the royalty go to the authors he used, and the rest to the author's fund?

BOUNCE-THROUGH ROYALTIES

When you use somebody's directory, you bounce through their specification to another document. What royalty goes where? (We want to encourage the creation of directories, so these authors should be rewarded.)

However, we also want to keep royalty a fixed rate.

One solution: transmit the full address of the desired document, which would yield a royalty proportional to the address length to the directory owner. He could even increase the incentive by increasing artificially the length of this transmission. But this reduces the capacity of the user's system by slowing him down.

MARKET-PRICING CONTROVERSY:
FIXED VS. VARIABLE

There are two schools of thought with respect to the pricing of these services. Surely the amount transmitted should not vary the price, since that would discourage high mental rates-- not what we want at all.

One school of thought has it that certain flat, predictable charges-- such as ten dollars an hour or two dollars an hour-- depending on class of service-- are the best way to go. We can call this "smorgasbord" pricing-- one price for all. It has the special advantage of avoiding hanky-panky in the accounting programs, which can then have conspicuous checksums. Further, the user can predict his overall expenses nicely. Perhaps most important, a uniform royalty for all authors and documents is also desirable because this means there is no pretext for the system's keeping track of who reads what.

Therefore, just as the postoffice subsidized the outlying stations on the basis of profits from the easy parts in the interests of uniform service at a uniform price, so might we.

Classical economics, however, suggests a more buoyant pricing mechanism, varying with time or system load-- "level-seeking," allowing market factors to enter in in a useful fashion.

1. Author Variations.
One such market factor would be to allow authors to set their own royalties-- very high, if they wanted. For now, in Balance I we have opted not to feature this.

2. Slack-time price float.
In this view, unused capacity should seek a "spot" market price, selling for less in short, or interruptible blocks.

3. Market price of disk.
In later stages, allowing rental of disk to be distributed among various vendors, with some market-pricing mechanism, is not out of the question.

(Variant proposals to hold costs constant by slowing down service have been proposed, as a method of allowing the pricing mechanism to enter the situation while maintaining constant cost-per-hour and, e.g., charging higher royalties for materials for a specific source. On the positive side, this allows pricing dynamics to operate and might allow users

4/5

to "break through" to full performance at a higher cost. On the negative side, it is philosophically most disagreeable.)

COST/TIME TRADEOFF

Cost and time are often a continuum. On our system, various areas of performance can be slowed down at lesser cost. In both behind-the-scenes and up-front ways, cost/time playoffs are important options in the tuning of the system.

THE RESOURCE UNIT

Users can ask for the moon and stars simultaneously. While early versions of the system will merely fetch what is asked for on a simple queuing basis, more sophisticated service algorithms will have to ration resources.

The Resource Unit (RU) then becomes a basic internal unit of software accounting, dividing the system's effort on your behalf. A standard customer gets one RU. (Priority customers might get more.)

If one entity is called for, the search for it proceeds with a force of one RU. If two entities are called for simultaneously, each gets 1/2 RU; and so on. RUs are divided as requests fan out.

It is easy to see why this is necessary. The request fanout can easily become astronomical, which is all right; the problem is to find an orderly basis for servicing MIRVed searches (Multiple Independent Reading Virtuality). The divisible Resource Unit keeps the overall Systems Effort equal to unity rather than inflating in combinatorial explosion.

ADVERTISING

The system does not discriminate in any way among "types" of document by content. Advertising is thus perforce allowed.

However, suggestions that advertising can somehow pay for generalized use of the system, as with TV and magazines, have pitfalls. Specifically, there is no foreseeable way to find out what is actually being shown on a screen; thus advertising could be automatically screened out in many ways, defeating the usefulness of it. So it is not clear that advertising subsidy is feasible.

4/6

ARCHIVING

It is mandatory that all sections of the service be profitable. This puts the question of "archiving" on a curious basis. Here the tradeoff between cost and time comes into sharpest relief.

DISK STORAGE, on a unifying hypertext system, can be effectively instantaneous. But disk storage can last as long as somebody pays for it. After that you go to tape.

THE TAPE PROBLEM

What is on tape takes longer to get to. And bringing in materials for tape takes lots of time-- time that can worsen drastically as demand escalates. (In the degenerate case of queuing, of course, we go to repeated Grand Passes of the corpus.)

Immense automatic tape systems are available, with little locomotives that go to the appropriate rack of tapes, pluck the one desired, slurp it in and put it back. But the maintenance of a large-scale tape library is expensive-- though less, of course, than disk per unit stored.

TWO SCHOOLS OF THOUGHT ABOUT TAPE

One view is that disk and tape should be a unified whole, with all that is on tape, an indefinitely expanding bundle, united to what is on disk; though subject to unpredictable delays.

However, this creates financial difficulties. Most of what stays on tape is no longer being paid for or referred to. Storing the tape is cheap; hooking it up is expensive. Finding a viable economic arrangement is the key problem.

One solution, and a probable one, is simply a surcharge for tape fetches.

(This in addition to the delay, which is an implicit charge.)

ARCHIVING ARRANGEMENTS

There may, after a point, have to be a charge for long-term tape storage-- or for guaranteeing it. Our provisional solution is to define service arrangements, and the organizations to maintain them, rather like "perpetual care" mortuary arrangements. This will be discussed later.

BALANCE I

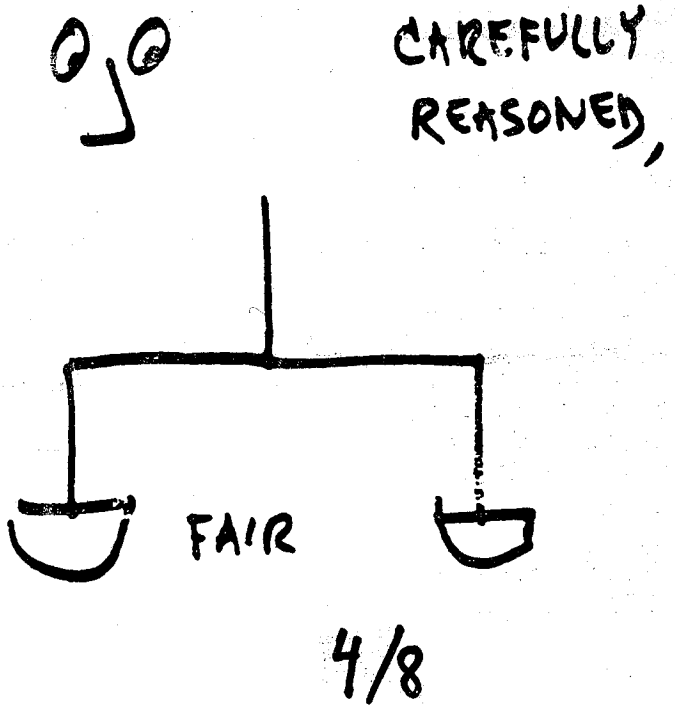
The overall scheme of incentives, this particular tuning, I call Balance I. It consists of the following provisions:

Two simple categories of privacy (published and private-- with private materials recallable, published requiring 6-month depublication notice. "Privashing" recallable, unlimited distribution, no royalty.

Fixed royalty, 5% of hourly charge, so that the computation of royalty is simpl e (to avoid hankypanky in the accounting).

Fixed charge by hour, not by amount transmitted. (We want to encourage people to read a lot, not to reduce their intake!)

Note that this arrangement is fair, orderly and simple. And these seem to me very important features.



4/8

If documents are used that have nobody for a royalty to go to, such as Shakespeare (or the use of your own private documents), the same amount goes to the Authors' Fund. A blue-ribbon panel assigns these proceeds to such purposes as typing in Persian poets or subsidizing writers.

Royalty to publishers of directories is proportional to the coordinate-material they transmit.

THE QUESTION OF SPECIAL SERVICES

In addition to the standard services, we could get into additional speedups within our philosophy (look-ahead, faster copies of specific versions). Or we could imitate other services by adding admittedly-useful features that they have and we don't.

Consider scans. Many retrieval systems have fulltext scans for keywords. In doing that we are intrinsically less efficient than anyone else, since they store blocks and we store fragments. However, by storing concordances we could effectively achieve the same searches. Ideally these would be concordances built concurrently with input typing. These would enlarge storage for a given document 1/5 to 1/3, however.

4/9

But we must always remember that our specialties are the rapid delivery of linked and windowing documents, and the assimilation and storage of changes. If we go far afield from this, our system could lose its power and ideals.

THE TRADEMARKS

The following are the trade and service marks of the system described in this book, by which we distinguish our information services and products from others.

"XANADU™" to denote all our information services and products.

"SilverStands™" to denote the stations at which service will be provided.

"FREND™" as an authorized and approved FRont-END program or console.

"XANAMAIL™" to denote personal message services.

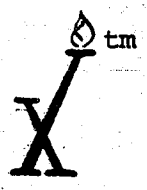
"XANACARE™" to denote arrangements for guaranteed long-germ storage.

"XANADOODLE" for computer graphic systems.

The slogans
"Lightning Literature™"
"The World of You™"
"The Wings of Mind™"
"Anything Instantly™"
"One Sky, One System™"

The following cartoon characters:
PORLOCK™
ROSEBUD™
XAN MAN™
The MARGINALIEN™
and
THE HOBGOBLIN OF LITTLE MINDS.™

And finally that X-ternal Device,
The Eternal Flaming X
in all its variants.



TUMBLING THROUGH THE DOCUVERSE

Our kingdom is already twice the size of Spain,
and every day we drift makes it bigger.

The Kaiser
in Herzog's film
Aguirre, The Wrath of God

Besides the actual contents of our system-- text, graphical data, and other notations representing things people want to look at and manipulate -- the system must keep track of a lot of numbers. These are the internal numbers that are used for counts and pointers, and the overall scheme of where things are and how to get to them. They are integers.* Some of these numbers have to be very very big. Others (in fact most of them) are small.

Our universe of documents (or docuverse) is potentially very large, and will grow unpredictably. Numerical addresses in our system can therefore grow very large. But they must also work with small increments and offsets. Designing the address space and notational representation is therefore crucial and difficult.

It is not obvious-- it was certainly not obvious to us at the outset -- how to specify such a universe in any tractable form, with an indexing scheme that can possibly grow very large and still retain any cogent manipulability when we deal with the nitty-gritty small increments of changes within a given document.

One assumption would treat the docuverse as a large integer domain, sparsely occupied by assigned document addresses. That way lies madness: it would mean unoccupied areas would use up many, many precious bits.

---

* Except where floating-point and trigonometric functions are used for certain proprietary algorithms.

We drew some inspiration from the Dewey Decimal system, which, despite its faults, does not waste a lot of space on empty characters. This leads to insights about forking numbers, which we have developed in an unusual way.

There are many kinds of numbers and notations for them. While it is customary in computer work to use several kinds of numbers (integers and floating-point numbers of different lengths, ASCII and BCD decimal trains), we use none of these in our current system design. For the interested reader, the types of numbers we have chosen to use are an interesting exercise in notational engineering.

Our solution has two parts. One is to use an accordion-like integer notation whose numbers are very short in representation when small, and as large as they need to be when big.

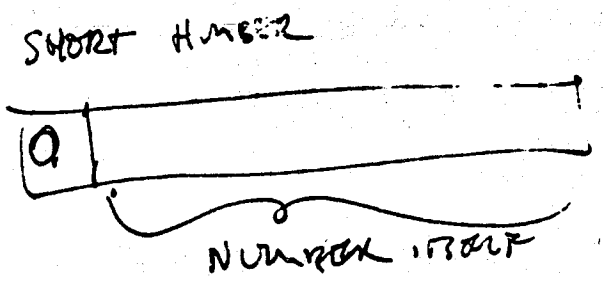
The second part of the solution is to define an accordion-like master address space, potentially very very large, which includes notational provisions for the complex relations between documents, their forebears, their owners, their locale, and the expansion of the network itself.

HUMBERS
(Variable-Length Binary Integers)

Humber stands for "Huffman-encoded number," which (strictly speaking) it is not; so if you prefer it stands for "humungous number."

Consider a byte.

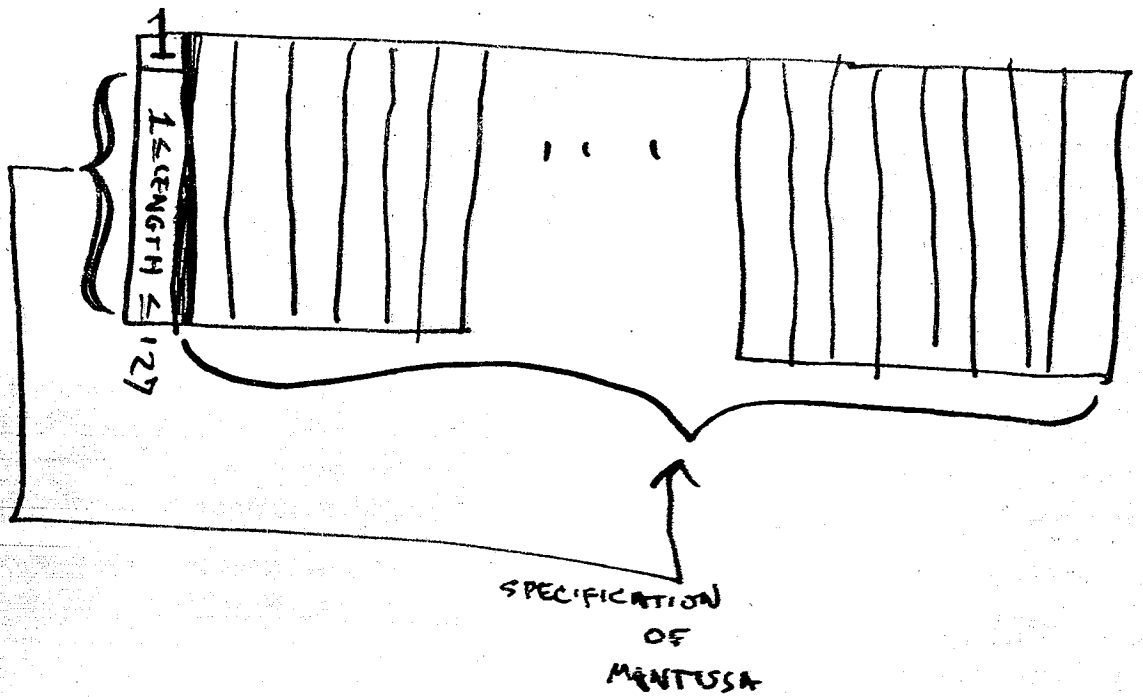
The first bit signals whether the number is complete in this byte. If this bit is unset, or zero, the remaining seven bits hold the number itself (Ø to 127), and the entire number is stored in the one byte.



If the Completeness bit is set, or one, that means the remaining bits of this byte specify the length, in bytes, of the number, in binary. Thus the humber may range up to $2^{127 \times 8}$, a number larger than needed very soon.

4/12

LONG NUMBER



4/13

Note, then, several advantages of this scheme. Small incremental humbers are one byte long. But very large humbers adhere to the same format; only one set of "humber arithmetic" routines is necessary.

It will be noted that these numbers occupy no more space than they need; they are short most of the time (when needed for small incrementation) and stretch out whenever needed without any change in the generalized manipulation routines. No more than seven bits are wasted in the length of the mantissa, and there is only the one-byte overhead of the specifier.

The Containment bit is zero if the actual number is within the byte, 1 if it is not; this choice makes an all-zero byte a true zero (a fact which will be seen to be a useful choice for the tumbler mechanism, below).

The Master Address Space:
TUMBLERS
Forking Multipart Integer Vectors, with Carry

The larger scheme for addressing in the docuverse, our present one, employs a multi-part number with some rather remarkable features. It is intended to keep track of hereditary successions of various kinds, while reducing the overall indexing manipulations to tractable arithmetic form. We call it a tumbler.

We chose the word "tumbler" partly because it sounded like "number" and "humber," and partly because of its curious relation to the rotary mechanisms of locks, which also slide with respect to one another, and are also called "tumblers."

The diabolical simplifying assumption that we have made is that there is really only one document.

A hypothesis built into the scheme is the notion that the number of compatible nodes will grow indefinitely but in hard-to-predict patterns.

Thus a node, or station, is seen as having ancestors and possible descendants. An account, too, and a document, all have possible descendants.

For instance, consider that you have written the twentieth document, #20, on a given node. Now you do a Version Fork, leaving you two versions

which you choose to designate as separate versions, the original number being superseded.* Now these two are versions 20.1 and 20.2 (while the parent document, 20, in fact continues to hold most of the contents of both). Now suppose you do another version fork on 20.2. This yields 20.2.1 and 20.2.2-- and so on.

The entire tumbler works like that: nodes can spin off nodes, accounts can spin off accounts; nodes can spin off accounts and documents; and so on. Thus all numeration in the docuverse is comprised to a single mechanism.

The tumbler format is:

NODE NO.	ACCOUNT NO.	DOCUMENT NO.	VERSION NO.	POSITION in version

Each of these fields may have one or more parts.

Two different field separators are required. Presently we use the decimal point for hereditary junctures within a given section; and the number Ø (between decimal points) to separate the major sections. Thus a tumbler might look like:

Ø.Ø.Ø.7.3.2.Ø.335.896

(The fields missing between the first three zeroes show this to be an incremental tumbler.)

A fuller specification of the tumbler is as follows:

H.Ø.H.Ø.H.Ø.H.Ø.H

where "H" is any hereditary multihumber (series of humbers representing hereditary segmentation by decimal points as described above).

Thus a large tumbler might look like this:

i...i.Ø.i...i.Ø.i...i.Ø.i...i.Ø.i...i

where "i" is any integer, represented how you like.

---

* Whether a parent version number continues on to evolve as one of the daughter versions is a user choice, and thus a front-end function.

4/15

(Note that we have skipped over the notational reconcilation between humbers and decimal points. If properly "understood," the decimal points can be simply left out.)

There is not time at the present writing to explain the rules of tumbler arithmetic as worked out by the group. Suffice it to say that tumbler addition is non-commutative (A+B does not equal B+A) and therefore there are strong and weak forms of subtraction (given A+B, both A-B and B-A).

It will be seen that the tumbler (and its associated routines of addition and subtraction) provides a master scheme for the full address space while handling increments and offsets -- whether local or very large-- with creditable brevity. These increments, and offsets, naturally, can cross the lines between nodes and accounts, documents and versions. So it's all really one big forking document.

Note that "time" is not included in the tumbler. This results in part from the interesting hypothesis that at some future time, document nodes may be in starships nearing the speed of light, so that their time records will not transform directly to those kept at stable locations. Time is kept track of differently.

Note also that our demonstration system uses standard integers as tumbler fields, not humbers. However, our coding is modulzrized in anticipation of this upgrade.

Humbers are the work of Roger Gregory, Mark Miller and Stuart Greene, done in the summer of 1979. While it went through many changes, and represents contributions by numerous members of the Xanadu troupe, the present form of tumbler was worked out by Mark S. Miller, with help from Roland King and Roger Gregory, in approximately June of 1980.

GENERAL REMARKS

The docuverse is the occupied address-space. We do not waste numerical positions on what is not there. As with Dewey Decimal, conceptual holes do not become utterly inefficient notational holes.

FEEL FREE

No proprietariety is asserted for humbler and tumbler methods or for their names; use them freely. (Indeed, they are a required part of the front ends.) However, the group would not mind a little credit now and then.

4/16

seded by a wieldier format when the system has been stabilized.

Apr 15 20:15 1981 xuinput.bnf Page 1

THE PROTOCOLS

CREATENEDOCUMENT

creates an empty document, returns docid.

INSERT <docid> <docvsa> <text>

<docid> := <tumbler>
<docvsa> := <tumbler>
<text> := <char>*

puts given text in document at given address, vstream addresses of following characters, if any, are increased by length of inserted text.

DELETEVSPAN <vspec>

<vspec> := <docid> <spanset>
<spanset> := *
 := <tumbler> <tumbler>

removes given spans from given document.

REARRANGE <docid> <cutset>

<cutset> := <ncuts> <docvsa>*
<ncuts> := <integer> /* ncuts = 3 or 4 */

cutset consists of 3 or 4 vsas within given document -- in 3-cut case material between 1st and 2nd cuts is interchanged with that between 2nd and 3rd cuts, in 4-cut case material between 1st and 2nd is interchanged with that between 3rd and 4th.

COPY <docid> <docvsa> <specset>

<specset> := <spec>*
<nspecs> := <integer>
<spec> := <vspec> |

material determined by specset (given vspans of given documents) is copied to document determined by <docid> at address determined by <docvsa>.

4/17

CREATENEWVERSION <docid>

created new document with contents of given document, returns new docid.

MAKELINK <docid> <docvsa> <fromset> <toset>

<fromset> := <toset> := <spanset>

creates link in given document at given address from fromset to toset.

RETRIEVEV <specset>

returns material (text and links) determined by specset.

RETRIEVEDOCVSPAN <docid>

returns a span determining the origin and extent of the vstream of the given document

RETRIEVEDOCVSPANSET <docid>

returns a spanset determining all sections of the vstream of the given document corresponding to distinct ispans

FINDDOCSCONTAINING <vspecset>

<vspecset> := <vspec>*

returns a list of all documents containing any of the material determined by the given vspecset

FINDLINKSFROMTO <homeset> <fromset> <toset>

<homeset> := <specset>

returns a list of all links which are (1) in homeset, (2) from all or any part of fromset and (3) to all or any part of toset

FINDNUMOFLINKSFROMTO <homeset> <fromset> <toset>

returns the number of links which are (1) in homeset; (2) from all or any part of fromset and (3) to all or any part of toset

FINDNEXTNLINKSFROMTO <specset> <fromset> <toset> <linkisa> <n>

<linkisa> := <tumbler>
<n> := <integer>

returns a list of all links which are (1) in the list determined by homeset, fromset, and toset, as in FINDLINKSFROMTO, (2) past the link given by <linkisa> on that list and (3) no more than <n> items past that link on that list

4/18

FINDNEXTNLINKSIN <vspec> <linkisa> <n>

returns a list of up to <n> links which are (1) in the indicated portions

of the document determined by <vspec> and (2) past the link determined by <linkisa> in the vstream order of their home document

> SHOWCORRESPONDANCES

> FINDLINKCONTENTCRESTRICTEDTOVSPECS <linkisa> <vspecset>

> NAVIGATEONHISTORICALTRACE

4/19