DOCUMENTATION FOR THE

NIST/EPA/NIH MASS SPECTRAL LIBRARY AND MS/MS PEPTIDE LIBRARIES

DLL VERSION 2.1.5.x

FOR MICROSOFT® WINDOWS®

Programmers Reference
April 2013

Dmitrii Tchekhovskoi, 221/A111
NIST Mass Spectrometry Data Center
National Institute of Standards and Technology
100 Bureau Drive, Stop 8320
Gaithersburg, MD 20899-8320

TEL: (301) 975-4673
FAX: 301-975-2643
e-mail: dmitrii.tchekhovskoi@nist.gov

Microsoft and Windows are registered trademarks of Microsoft Corporation

Portions copyright 1984-1996 FairCom Corporation. "FairCom" and "c-tree Plus" are trademarks of FairCom Corporation and are registered in the United States and other countries. All Rights Reserved

zlib software copyright 1995-2010 Jean-loup Gailly and Mark Adler (http://zlib.net)

.

Changes as compared to the previous version are in purple color; new items in the headings and Contents are marked with asterisks *

CONTENTS

TECHNICAL OVERVIEW 6
Redistributables and Import Libraries 7
NEW FEATURES SUMMARY 8
DLL FUNCTIONS 9
I. Initiation 10
II. Termination 10
III. Single Spectrum Search 10
IV. Searching For a Set of Spectra 11
1. Chemical Formula 11
2. Molecular Weight 11
3. Chemical Abstracts Service Registry Number 11
4. A range of Library ID Numbers 11
5. A range of NIST Registry Numbers* 11
6.1. Exact Molecular Mass* 12
6.2. (Re)indexing exact molecular mass in a library* 13
V. Incremental Name Search 13
VI. Spectra with Specified Peaks 15
VII. Standard EI Library Search 16
1. Identity Search 16
2. Similarity Search 16
3. Neutral Loss Search. 16
4. Hybrid Search. 16
Screening and Comparison 16
VII-A. No Presearch Library Search 17
VII-B. MS/MS Library Search (NIST MS dll version 2.1.1 and later) 17
VIII. Sequential Search 18
IX. Data Retrieval 19
1. Main Spectral Data 19
2. Synonyms 19
2.1 Tagged Synonyms * 19
3. Contributors and Comments 19
4. Replicate Spectra 19
5. Chemical Structures 19
IX-A. Handling Chemical Structures 20
X. Constraints and Hit List Utility 22
XI. Maintaining User Libraries 22
XII. Molecular Weight Estimation 23
XIII. Molecular Weight Estimation 2 23
XIV. Chlorine - Bromine Estimation. 23
XV. Substructure Information. 24
XVI. Creating New User Libraries 24
XVII. Checking Library Type and Size 25
A. GetLibraryType Function 25

B. GetNumberOfEntries Function 25
C. Determining Search Types Available for a Library 25
D. Library Files Responsible for Various Search Types 27
DATA STRUCTURES 28
NISTMS_IO 28
NISTMS_RECLOC (2 GB API only) 29
NISTMS_MASS_SPECTRUM 30
NISTMS_SRCH_CONTROLS 31
Search mode flag 32
NIST MS Search Limits Minimum m/z vs. cntls->min_mass 33
NIST MS Search Limits Maximum m/z vs. cntls->max_mass 33
NISTMS_CONSTRAINTS 34
NISTMS_STDATA 36
NISTMS_HIT_LIST 37
NISTMS_INC_NAME_INFO 38
NISTMS_PEAK_INFO 39
NISTMS_PEAK_INFO::ExactPeak * 40
NISTMS_AUX_DATA 41
INTERP_MS 42
ClBrStruct 42
NISTMS_USER_STRUCT_INFO 43
MISCELLANEOUS 44
Spectrum and Structure Pointers 44
User Library Creation 44
Naming Conventions 44
C/C++ Structure Alignment 44
Special Characters in Compound Names 44
Tagged Synonym Display 45
Tandem Mass Spectra Representation Conventions * 46
Precursor m/z * 46
(a) NIST 12 MS/MS Small Molecules Spectra * 46
(b) NIST 12 MS/MS Peptide Spectra * 47
(c) Spectra in NIST peptide libraries * 47
(d) In-source spectra * 47
Limitations 47
Calling Conventions 48
"SEARCH TYPE" SUMMARY TABLE 49
UPDATING FROM PRIOR VERSIONS 54
Breaking Changes in version 2.1.5* 54
Changes to NIST MS/MS 12 Incremental Name Search Information * 54
Changes in case of negative *min_mass* and *max_mass* (NISTMS_SRCH_CONTROLS) 54
New Features in version 2.1.5* 55
New Features in version 2.1.3 55
New Features in version 2.1.2 55
New Features in version 2.1.1 55
New Features in version 2.0 56
Summary of Implementation Changes in versions 2.1.5* 56
Data Structure members in f32 format * 57
Summary of Implementation Changes in versions 2.1.3. 58

Summary of Implementation Changes in versions 2.1.2. 58
Summary of Implementation Changes in version 2.1.1 58
Summary of Implementation Changes in version 2.0 59
Bugs Fixed in version 2.1.5 * 60
Bugs Fixed in version 2.1.3, revision May 23, 2011 * 60
Bugs Fixed in version 2.1.3 60
Bugs Fixed in version 2.0 61
How to update to version 2.1.5 * 62
How to update to 2GB API included in version 2.1.3 62
How to update to version 2.1.3 * 62
How to update to version 2.0 63
Differences between v. 1.5 and v.2.0 user libraries 63
APPENDIX 63
1. Syntax of Name Fragments Constraint 63
2. Syntax of Tags in Comment and Text Info Constraint 64
2.1. General 64
2.1.1. Tags in Spectrum Text Information * 64
2.2. Entering Tags in Comment Constraint 65
2.2.1 Examples 65
2.2.2 More formal description 66
2.2.3 More formal description of reversed constraints 67
2.3. Entering Tags in Text Info Constraint * 67
2.3.1. Examples * 67
2.3.2. List of acceptable special tags * 68
3. Syntax of Peptide Sequence Constraint 68
3.1. Alternatives to using dots in Peptide Sequence Constraint 68
3.2. Examples 69
3.3. Abbreviations in Peptide Sequence Constraint 69
4. NISTMS_RECLOC type members of 2GB API 69
5. 2GB API NISTMS_RECLOC ftype values 70
5. f32, a 32-bit Mass and m/z Floating Point Representation (version 2.1.5)* 71

TECHNICAL OVERVIEW

This NIST MS DLL provides a comprehensive set of tools for locating and retrieving data in mass spectral reference libraries from Microsoft® Windows® 32-bit and 64-bit applications. Most calls are made through a two-parameter function:

nistms_search(NISTMS_SEARCH_TYPE *search_type*, NISTMS_IO **nistms_io*)

The argument *search_type* defines the action to be taken and *nistms_io* provides buffers for input and output data. The variable *search_type* is an enumerated variable of type NISTMS_SEARCH_TYPE and *nistms_io* is a pointer to the principal data structure, NISTMS_IO. The active data elements in *nistms_io* depend on *search_type*.

All data types and related definitions are supplied in an annotated header file, NISTMS.H, which must be included in any code that accesses *nistms_search()* or uses any of the structures referenced in *nistms_io*. General features of the DLL are discussed below.

ACCESSING DATA: Library searches return locations of spectra. Spectral data and related information are obtained by submitting these locations in separate calls. To assist in the rapid construction of hit lists for user display, in some searches certain identification information may optionally be returned along with the spectra locations.

RELATIONS BETWEEN CALLS: Most calls to *nistms_search*() act independently of one another. However, because of the need to provide the user with intermediate results, results from some calls implementing the "anypeaks" search, sequential search, and the incremental name search depend on previous calls.

ACTIVE LIBRARY: A library is a collection of files with predefined names and formats located in a separate directory. All libraries to be accessed must be passed in an initialization call. Most subsequent searches specify which of these libraries to use (active libraries). Most searches capable of returning multiple spectra can also access multiple libraries.

LIBRARY TYPES: Three varieties of libraries are supported, 1) the main NIST/EPA/NIH collection (NISTMS_MAIN_LIB), 2) the NIST selected replicate spectra library (NISTMS_REP_LIB), and 3) user-created and maintained libraries (NISTMS_USER_LIB). The Replicate library is unique, and may be used only in spectrum and ID number searches. Substance-oriented searches (formula, CAS number, molecular weight) for NIST spectra must retrieve replicate spectra locations from the corresponding NIST main library entry. User libraries are just like the NIST main library except that they can be edited and cannot contain more than 65,535 spectra each. If a compound from the user library has a CAS number, but does not have structure or alternative chemical names (synonyms), the software retrieves these data from the NIST main library if possible.

MEMORY ALLOCATION: The calling program is responsible for allocation of all buffers passed to the DLL. When required, lengths of variable length buffers are passed along with pointers to these buffers. For optionally returned data, passing a NULL pointer in the appropriate buffer pointer signals that this data should be not returned.

ERROR HANDLING: Most errors and warnings are returned as a non-zero *error_code* value in NISTMS_IO, with appropriate *#defines* given in NISTERR.H. Certain errors, mostly dealing with user library maintenance, are handled as soon as they are encountered.

USER LIBRARIES[1]: Libraries containing spectra and chemical structures specified by the user may be created, maintained and searched. A new, empty library can be created with the CreateUserLibrary() function. GetLibraryType() returns the type of user library at a specified location.

Redistributables and Import Libraries

	Regular (¼GB) API	**New 2GB API**
Max. MS library file size supported	268,435,455 bytes	2,147,483,647 bytes
Max. number of libraries	16	127
Supported library types	EI and MS/MS	EI and MS/MS
. dll version	2.1.5.x	2.1.5.x
Microsoft Visual Studio 2008/2010 solution configuration name	Release_peptides	release_pep_2gb
C/C++ preprocessor definitions required for NISTMS.H header file (x86 and x64 platforms)	WIN32 ~~INTERNALBUILD~~	WIN32 ~~INTERNALBUILD~~ NISTMS_6BYTE_RECLOC
Redistributable binaries, x86	nistdl32.dll ctNT66.dll zlib1.dll	nistdl32_2gb.dll ctNT66.dll zlib1.dll
Import library, x86	nistdl32.lib	nistdl32_2gb.lib
Redistributable pre-built .NET interface assembly, x86	nistmsclp.dll	nistmsclp_2gb.dll
Redistributable binaries, x64 *(beta version)*	nistdl64.dll ctNT66_64.dll zlib1_x64.dll	nistdl64_2gb.dll ctNT66_64.dll zlib1_x64.dll
Import library, x64 *(beta version)*	nistdl64.lib	nistdl64_2gb.lib
Redistributable pre-built .NET interface assembly, x64 *(beta version)*	nistmsclP.dll	nistmsclP_2gb.dll
.NET interface assembly namespace	NISTMSCL	NISTMSCL2
.NET Framework version of the pre-built .NET interface assembly	Microsoft Visual Studio 2008 (VC9): 2.0 Microsoft Visual Studio 2010 (VC10): 4.0 Microsoft Visual Studio 2012 (VC11): 4.5	

Note 1. Currently, **x64** binaries are in **beta** version: they have been subject only to limited testing.
Note 2. x64 version of nistmsclP.dll intentionally has the same name as x86 version, except letter case. As the result, the same "ANY CPU" build of a .NET application should be able to work with the correctly installed version of the .NET interface assembly both under x86 and x64 versions of Windows.

[1] User libraries created with the NIST MS Dll can be edited with the NIST MS Dll and cannot contain more than 65,535 spectra each. A special read-only type of user libraries may contain hundreds of thousand spectra. NIST MS Dll can search such libraries but cannot create or edit them.

Note 3. .NET interface assembly NISTMSCL is a "strong name assembly", culture="neutral" publicKeyToken="1cdbd1c0f2a1aa22".

NEW FEATURES SUMMARY

Several new features have been added since the January 1999 version of the DLL for handling user-created MS libraries.

STRUCTURES IN USER LIBRARIES. Starting from the January 1999 release, DLL can add and retrieve structures from the user-created MS libraries as well as scan structures in the user libraries and user-supplied files. A one-time procedure for indexing structures in older user libraries is provided.

INCREMENTAL NAME SEARCH. Incremental name searching can be performed for user-created libraries. A one-time procedure for adding the feature to older user libraries is also provided, as well as backward compatibility.

ADDING/DELETION OF SPECTRA TO THE USER LIBRARIES. This process is now much faster at the expense of minor changes in the user libraries format. While older user libraries are fully compatible with this software; user libraries created or extensively changed with this software may not be compatible with older releases of the NIST DLL or with MS Search Program ver. 1.6x and earlier.

SUPPORT FOR CHEMICAL NAME SYNONYMS IN USER LIBRARIES. Starting from the August 1999 release, DLL can add and retrieve chemical name synonyms from the user-created MS libraries. Presence of the synonyms in the user library may render it incompatible with versions of the NIST MS Search prior to v1.7.

NO PRESEARCH SEARCH. Starting from the September 1999 release, a new type of user spectrum search was added. It compares the unknown spectrum to all spectra in the active libraries. The hit list produced by this search may contain up to 400 spectra.

SUPPORT FOR COMMENT/CONTRIBUTOR FIELD SEARCHES. Starting from the June 2006 release (version 2.1.1), the maximum size of the Comment/Contributor field, NISTMS_MAXCONTRIBLEN, was increased to 2048 bytes. The DLL accepts constraints (conditions) for Constrained spectrum search and Sequential search that allow selection of only those spectra whose Comment/Contributor field satisfy the specified constraints. For details on this type of constraints see “Syntax of Tags in Comment and Text Info Constraint” section in the Appendix.

Peptide MS/MS LIBRARY SEARCH. Starting from the June 2006 release (version 2.1.1), a search designed specifically for matching MS/MS fragmentation spectra was added along with a number of peptide search-specific constraints. These search and constraints may be used only for searching a MS/MS spectrum in MS/MS libraries. Unlike an EI spectrum, a MS/MS spectrum always has precursor ion m/z value, which may be non-integer as well as peak m/z and intensities. In a MS/MS spectrum, each peak may be accompanied by a text annotation; these annotations are used in

searching peptide MS/MS libraries prepared at NIST[2]. The peptide-specific constraints may be used only in Sequential search and No presearch MS/MS search.

LARGE LIBRARY SUPPORT. Starting from June 2010 release (version 2.1.3) the DLL supports searching libraries that have up to 1,048,560 spectra per library (previous versions could search libraries with up to 786,420 spectra per library);

2GB API VERSION. Starting from June 2010 release, version 2.1.3 of nistdl32_2gb.dll has a new 2GB API, which allows searching libraries that have file sizes up to 2,147,483,647 bytes. The larger file size is supported by replacing a 4-byte library/file offset of type "long" with a 6-byte type "NISTMS_RECLOC". The regular API version (also called ¼GB API) exported from nistdl32.dll supports file size up to 268,435,455 bytes; this dll is retained for compatibility reasons. In the regular API, NISTMS_RECLOC is defined as long int.

Small Molecules MS/MS LIBRARY SEARCH (April 2013 release). This search does not use peptide-specific product peak annotations and other peptide-specific tuning. It also has an improved ability to identify compounds with a few dominant peaks – a common situation for Tandem MS.

ALTERNATIVE PEAK MATCHING METHOD (April 2013 release) It improves the reliability of the score when searching noisy MS/MS spectra.

UP TO 127 LIBRARIES. (April 2013 release) Version 2.1.5 of nistdl32_2gb.dll (2GB API) supports initiation and searching of up to 127 libraries simultaneously.

EXACT MASS (April 2013 release) Search, Constraint, and Any Peaks search Exact Mass peak search.

IN-SOURCE TANDEM SPECTRA (April 2013 release) Spectra with accurate peak m/z and intensities, which do not have a well-defined precursor m/z (to search for them, use no presearch MS/MS search)

COMPATIBILITY WITH PREVIOUS VESRIONS. To make the introduced in v. 2.1.5 new API described in this document compatible with software designed for the previous version, 2.0, of the DLL the new features are turned off by default. The initiation of the version 2.1.5 features turns them on (see Initiation section below.) However, the users are advised to avoid using ver. 2.1.3 and lower compatibility mode because in this release it has not been tested.

DLL FUNCTIONS

Annotated "C" code illustrating each type of search is given in CALLDLL.C. An overview of these functions, grouped according to general type, is presented below.

[2] See http://peptide.nist.gov

I. Initiation

The opening call to the DLL provides paths to all libraries which may be accessed later. In this call, *search_type* = NISTMS_INIT_SRCH and the NISTMS_IO structure must contain paths to available library files (*lib_paths*), along with their types (*lib_types*). Subsequent NISTMS_INIT_SRCH calls may invalidate previous spectrum and structure locations if the order of the libraries in *lib_paths* has changed.

The DLL expects work directory path to be specified along with library paths. The work directory is a directory where the NIST MS DLL stores temporary files. Not every search uses the directory, only some complicated user library searches need it. The work directory should exist and be specified in *work_dir_path*. If the work directory path is not specified the DLL will display assertion warning. If the path is specified but the directory does not exist some user library searches will not work properly. If several applications use the NIST MS DLL at the same time they must have different work directories.

To make the API described in this document compatible with software designed for the previous versions (2.0.0.x) of the DLL the new features available since v.2.1.1 are turned off (disabled) by default. The NIST MS dll needs special initiation to enable these features, namely, MS/MS spectrum search, MS/MS-specific features, peptide-specific features, exact mass features, and comment/contributor constraints: *search_type* = NISTMS_SET_VERSION while *io->string_in* points to the version string, "2.1.5" [3]. To disable all features use version string "2.0".
It is sufficient to initiate the new features one time, right after the DLL is loaded by the application.

II. Termination

The last call (*search_type* = NISTMS_CLOSE_SRCH) simply frees memory and closes library files. This procedure is automatically called prior to a reinitialization (NISTMS_INIT_SRCH).

III. Single Spectrum Search

1. Library ID Number (*search_type* = NISTMS_ID_SRCH):
This requires an ASCII NULL-terminated string containing an ID number and is intended to retrieve a single spectrum location in a specified in *active_libs*[0] library. (Each spectrum in a given library has a unique positive identification number. The ID number itself will generally be acquired as a binary integer from an earlier search, hence it must be converted to ASCII for this search). A pointer to the input string is attached to *string_in* (type char* in the NISTMS_IO structure). If a retrieval is found, its location is returned in *output_spec_loc* (type NISTMS_RECLOC in the NISTMS_IO structure). If no spectrum is found, *output_spec_loc* is zero. Only one library may be searched for Library ID Number at a time. See IV.4 for an extended version of this search.

2. Backward compatibility note. Since Incremental Name Search (see below) is now permitted for user libraries, the Full Chemical Name search implemented in previous to 1999 release of the DLL should be considered obsolete and is kept only for backward compatibility.

[3] To avoid breaking changes, use version string "2.1.4" string instead of "2.1.5". However, this may reduce precursor m/z accuracy retrieved from a library spectrum from 10 ppb to 0.01 m/z units.

IV. Searching For a Set of Spectra

As when searching for a single spectrum, input specifications are passed as an ASCII string, *string_in*, in the NISTMS_IO structure. More than one library may be specified for searching in *active_libs*. Spectra locations are returned in the field *spec_locs* (type NISTMS_RECLOC*) in a NISTMS_HIT_LIST structure with the number of hits returned in *num_hits_found.*

1. Chemical Formula (*search_type* = NISTMS_FORMULA_SRCH)
A conventional chemical formula provides the input for this search. Elements may be in any order, but the first character of each element should be upper case and the second character should be lower case. Input may also be provided as a semi-structural formula (e.g, C6H3(CH3)2(C4H9)), although only the net formula will be used in searching (C12H18, in this case).

2. Molecular Weight (*search_type* = NISTMS_MW_SRCH)
This is a "nominal" molecular weight based on integral molecular weight of the most abundant isotope of each element.

3. Chemical Abstracts Service Registry Number (*search_type* = NISTMS_CASNO_SRCH and NISTMS_CASNO_SRCH2[4]) :
This ASCII number may be submitted with or without conventional hyphens.
NISTMS_CASNO_SRCH2 also searches among Related and Salt/Mix CAS r.n. first introduced in NIST 11 MS Library.

The NIST replicate library cannot be directly accessed in these three searches. Locations of replicate spectra are optionally given along with the data for the corresponding compound in the main library.

4. A range of Library ID Numbers (*search_type* = NISTMS_ID_SRCH):
This search can access any single library specified in *active_libs*[0], including NIST replicate library. Input range of IDs is passed as an ASCII string, *string_in*, in the NISTMS_IO structure; for example "1-100".

5. A range of NIST Registry Numbers* (*search_type* = NISTMS_NISTNO_SRCH[5]):
This search can access all libraries specified in *active_libs*[], including NIST replicate library. Input range of NIST Registry Numbers is passed as an ASCII string, *string_in*, in the NISTMS_IO structure; for example "1-100". Previously to NIST 12 MS/MS Library, this search could be used to search only NIST Main and Replicate libraries.

[4] This search uses one of index files registry2.in6 or registry2.inu if it is present in the library folder. Such libraries can be identified with NISTMS_MARK_LIBS or NISTMS_MARK_ALL_LIBS searches. If registry.in? file is not present, the search uses registry.in6 or registry.inu, which are used by NISTMS_CASNO_SRCH. Currently, "registry2.in?" files cannot be created with NIST MS Search, NIST MS DLL, or Lib2NIST.
[5] NIST Registry Number search uses one of index files specno.in6, specno.inr, or specno.inu if it is present in the library folder. The last file is used in NIST user libraries starting from NIST 12 nist_msms library. It may be created by Lib2NIST. Indexed by NIST Registry Numbers libraries can be identified with NISTMS_MARK_LIBS or NISTMS_MARK_ALL_LIBS searches.

6.1. Exact Molecular Mass* (*search_type* = NISTMS_EXACT_MASS_SRCH[6]):
This search can access all libraries specified in *active_libs*[], except NIST Replicate library. The requested exact mass range passed as an ASCII string, *string_in*, in the NISTMS_IO structure. The string contains items separated by space characters or semicolons. Each item starts with a tag (a single lowercase letter) followed by a value, which is a number or a chemical formula, with optional equal sign between them. Either **m** or **f** item must be present in the string.

tag	value	type
a	Absolute accuracy of exact mass in mmu or m/z in mmu/z	FPV
r	Relative accuracy of exact mass or m/z in ppm (parts per million)	FPV
f	Chemical formula to obtain the exact mass	fml
m	Exact mass or m/z (mass does not allow charge; m/z requires charge)	FPV
l	Loss, chemical formula (**l** is a lowercase letter L)	nfml
g	Gain, chemical formula	nfml
c	Charge (may be negative): interpret m as m/z	IV
i	Search among the given number of most abundant isotopes	NP
e	value=0 means neglect mass of the electron; value=1 is default	bool

FPV – nonnegative floating point value
IV – integer value ≠ 0
bool – integer value, 0 or 1
NP – integer value from 1 to 16
fml – chemical formula
nfml – chemical formula, possibly preceded by an integer number without space in between.

Chemical formulas may include parentheses; Loss and Gain formulas may be preceded by a positive integer coefficient, e.g. 2H2O.
If neither **a** nor **r** uncertainty is present, the uncertainty used is 5 in the next to the last digit of the entered mass or m/z. For example, the uncertainty for **m** =28.00 would be 0.005. In case of a chemical formula, missing uncertainty means the search calculates the same exact mass as was used for indexing.

Incompatible pairs of labels: (a,r), (f,m), (f,c).
Note: e is used only if both **m** and **c** are present.
It is assumed that the exact m/z was obtained from exact MW by adding the Gain, subtracting the Loss, and adding/removing electrons to set the Charge.

Examples
E1. Positive ion with m/z=111, charge=+2, measured m/z accuracy=33 ppm, was created by attaching $2H^+$ and removing H_2O; find possible neutral precursors, which have this monoisotopic peak:

"**r**=33; **m**=111; **g**=2H; **l**=H2O; **c**=2", or
"**r**33 **m**111 **g**2H **l**H2O **c**2"

Interpretation: Search in the library for substances with
exact mass = 111×2 – m(2H) + m(H_2O) + 2×m(electron);
in the mass range ±111×2×33×10^{-6} Da

[6] Exact Molecular Mass search uses one of index files exactmw.in6 or exactmw.inu. These files may be (re)created with *search_type* = NISTMS_INDEX_LIBRARY_EXACT_MASS.

E2. Positive ion with m/z=111, charge=+2, m/z accuracy=33 ppm, was created by attaching $2H^+$ and removing H_2O; find possible neutral precursors, which have this peak as one of 3 most abundant isotopes, including monoisotopic peak:

"**r**=33; **m**=111; **g**=2H; **l**=H2O; **c**=2; **i**=3", or
"**r**33 **m**111 **g**2H **l**H2O **c**2 **i**3"

Interpretation: Search in the library for substances, which have one of 3 most abundant peaks with exact isotopic or monoisotopic mass = 111×2 – m(2H) + m(H_2O) + 2×m(electron);
in the mass range ±111×2×33×10^{-6} Da

E3. Find all substances that have exact monoisotopic mass in the range ±0.03 Da (±30 mmu) around the exact mass of C_6H_6 with one H removed:

"**a**=30; **f**=C6H6; **l**=H"

Interpretation: Search for substances with mass = m(C_6H_5) = 77.03913 Da in the mass range ±30 mmu. This search is equivalent to the one with "**a**=30; **f**=C6H5"

Note 1. Loss (**l**) was added in examples E1 and E2 to obtain MW from m/z and subtracted from the exact mass of C_6H_6 in example E3.
Note 2. "**a**=30; **f**=C6H6; **l**=H2O" will produce an error because C6H6 does not have atom O.
Note 3. If Exact Molecular Mass index file, exactmw.inu, is not present in a library folder, it will not be created when a spectrum is added to the library. To enable indexing of the newly added spectra, re-index exact molecular mass in the library (see the next section, 6.2.)

6.2. (Re)indexing exact molecular mass in a library* (*search_type* = NISTMS_INDEX_LIBRARY_EXACT_MASS). To do this, make the library the only active library. To (re)index Mainlib, *string_in* must be "mainlib". Indexing requires a valid chemical formula. Spectra that have AUX_DATA::exact_mw and no valid chemical formula are not indexed.

V. Incremental Name Search

This search returns, in sorted order, names or name synonyms, library ID numbers, and sequence numbers specifying name position in the sorted order for compounds in the specified in *active_libs*[0] library (NIST Main or User Library). A special data structure, NISTMS_NAME_INFO, is used for this search. A specified number of names (*num_names_desired*) are searched for, and the number of names retrieved in *names* is returned as *num_names*. The array referenced by *id_nums* contains ID numbers. The array referenced by *name_pos* contains name position numbers (quite close to percentiles for large libraries); which are provided to support scroll bar positioning. The variable *last_name_pos* contains the name position number of the very last name in the library for the given sorted order; it is never greater than 101. Name string is referenced by *string_in* member of NISTMS_IO.

There are three ways of retrieving names:

1) Starting with the library name just before the submitted name string (*search_type* = NISTMS_INC_FIRST_NAME_SRCH);

2) Starting with the last library name having the specified name position number (*search_type* = NISTMS_INC_FIRST_NAME_SRCH, use special name string like "#23" where 23 is a name position number);

3) Starting with the nth name after (n > 0) or before (n < 0) the first name retrieved in the most recent incremental name call (*search_type* = NISTMS_INC_NEXT_NAME_SRCH; a special name string like "#23" or "#-23" is used for n = 23 or n = -23);

For any library, an incremental name search must begin with search type 1) or 2) above; after changing the content of the current (user) library, the search type 1) or 2) must also be the first invoked.
The "first" name referred to in 3) above is the first one returned in *names* by the preceding call.

Two name sorting schemes are available, one using only letters and ignoring prefixes (including Greek symbols) and numbers (*alpha_only*=1), the other ignoring only punctuation and other non-alphanumeric characters (*alpha_only*=0). Punctuation, including spaces, are ignored in both cases. Greek letters may be specified using their English word equivalent (alpha, beta, ...).

For faster response searches 1), 2), and 3) can be canceled using callback function described in section **VII,** Library Search.

The "name string" to be used in search 1) must be consistent with the *alpha_only* value. For *alpha_only*=1 it should contain only alphabetic characters (A-Z) and when *alpha_only*=0 it should contain only alphanumeric characters (A-Z, 0-9). Punctuation is always ignored. To create a name string out of the compound name, use *search_type* = NISTMS_INC_GET_NAME_KEY (see below). Only the first 18 characters of a name string are used for searching.

Returned ID numbers in *id_nums* may be used to retrieve data through the NISTMS_ID_SRCH (see **III**.1 above).

Only for the very last name in the library the returned position number is equal to *last_name_pos*; and only for the very first name in the library the returned number is equal to zero.

In search type 2), the name string "#0" retrieves names starting from the very first name; "#1" retrieves names starting from the name at about 1/(*last_name_pos-1)* fraction of names, "#2" -- at about 2/(*last_name_pos-1)* fraction of names, etc. "#(*last_name_pos-1)"* and greater retrieves only the very last name. ("#100" always retrieves the very last name)

search_type = NISTMS_INC_GET_NAME_KEY creates name string out of the compound name according to *alpha_only* value. The name string can be used for search type 1).
Since in case of sorted order *alpha_only*=1 a special (hard for the user to read) name string for searching is needed to reduce the ambiguity arising from the removal of all prefixes and digits, to create a name string out of a compound name for displaying use *alpha_only*=2.

search_type = NISTMS_INDEX_LIBRARY_NAMES is a special one-time procedure for allowing the incremental name search to be performed for older user libraries (those created with software released prior to 1999). It adds or replaces files responsible for incremental name search to the user library and does not change any other existing user library files. However, the library files should not be write protected. **Important:** Do not index names in NIST 12 MS/MS Libraries nist_msms and nist_msms2 because MS/MS information displayed in the Names window will be lost.

Backward Compatibility issues. If *name_pos* pointer is set to NULL or a number of active libraries is zero then the incremental name search will work the way it worked in versions of the NIST software released prior to 1999.

VI. Spectra with Specified Peaks

Spectra in one or more libraries having specified peaks may be rapidly retrieved in an interactive search. Four types of peaks are supported.

a) Normal: A conventional peak defined by an integral mass (m/z) and relative abundance window (% of largest peak in spectrum). The allowed abundance range is 1% to 100%.

b) Neutral Loss: Peaks RELATIVE to the molecular ion represented as a mass (m/z) LESS THAN the molecular ion, together with an abundance window (% abundance from 1% to 100%). For instance, mass = 0 denotes the molecular ion and mass = 15 is a methyl loss. Losses may be no greater than 64 m/z.

c) Rank: Normal mass (m/z) and range of allowed **ranks**. The rank of a peak is its position in a peak list ordered by descending abundance. For example, the largest peak has a rank of 1. Ranks up to 16 are allowed.

d) "Maxmass": This is designed to represent the most prominent high mass peak in a spectrum. No peaks at higher mass should have abundances above 5% or greater than the abundance of the specified peak. Abundances may fall between 1% and 100% of the largest peak in the spectrum.

e) Exact Mass[7]: This peak type is similar to Normal, but the mass is accurate. An accurate mass, an m/z value (adjusted for the electron mass), or an elemental composition of the ion (formula) can be entered. Only monoisotopic peaks may be searched. The abundance range is from 0 to 100% of the base peak. 0% means abundance < 5 for base peak intensity = 999.To fill out peak information, convert obtained elsewhere exact mass limits to f32 format by calling function nistms_dbl_to_f32(…) or use *search_type*= NISTMS_GET_EXACT_MASS_LIMITS to fill out *exact_mw_min* and *exact_mw_max* members of NISTMS_IO::NISTMS_PEAK_INFO::ExactPeak (only f, m, or m and c=1 options are allowed); also fill out *abmin* and *abmax* members of the same structure with appropriate values.

This search has been implemented in a way to permit close user interaction. Searching is initiated with *search_type* = NISTMS_ANYPEAK_INIT_SRCH. Each peak specification requires a separate call with *search_type* = NISTMS_ANYPEAK_ONE_PEAK_SRCH, which returns both the number of spectra having the newly entered peak along with number of spectra having different numbers of the peaks entered so far. After all peaks have been entered, a final call, *search_type* = NISTMS_GET_HITS_SRCH, retrieves locations of spectra having at least a specified minimum number of these peaks and frees allocated memory. If, for example, this specified minimum number is the same as the number of peaks entered, each retrieved spectrum will contain all specified peaks. A final call will fail if number of found spectra is greater than 6000 (defined as NISTMS_MAX_FPOS in NISTMS.H).

[7] Exact Mass peak type cannot be used in NISTMS_CONSTRAINTS.

VII. Standard EI Library Search

Four varieties of electron ionization library searching (spectral pattern matching) are available:

1. Identity Search. If the unknown compound is likely to be in the library, this is the quickest way to find it. It uses a very efficient method of selecting a small set of spectra for subsequent spectrum-by-spectrum comparison. Two speeds are available, **quick** and **normal**. Quick is more likely to miss distorted spectra, but is several times faster than normal, which will almost never fail to retrieve matching compounds. If results are not satisfactory, the unknown compound is probably not in the library and a similarity search should be tried. Another variation on **normal** is a search that applies penalties to the match factors depending on the rarity of the compound (number of "other databases" in which the CAS registry number is listed). The maximum penalty is 50 (out of 1000). This tends to position common compounds above exotic isomers that have nearly identical spectra in the hit list. It has no effect on spectra in user libraries.

2. Similarity Search. This search finds a larger set of spectra to compare with the submitted spectrum, so it will generally be slower than an Identity search. It was designed for retrieving similar compounds when the unknown is not in the library or its spectrum is distorted so badly that a reliable match is not possible.

3. Neutral Loss Search. If it is determined that the unknown compound is not in the library and the user (or your software) can estimate its molecular weight, this search will retrieve a set of compounds having the most similar "neutral loss spectra" (spectra measured as starting from the molecular ion). Certain classes of chemical structures have characteristic neutral loss spectra, and can often be identified by examining the chemical structures reported in the hit list. Hence, the most effective implementations of this (and the next) search will display chemical structures of compounds in the hit list.

4. Hybrid Search. This uses both conventional and neutral loss peaks in its search logic. As for the neutral loss search, an estimate of the molecular weight of the unknown is required. If the unknown compound contains chemical structures that generate both characteristic ions and neutral loss patterns, these structures can be identified from the hit list produced by this search.

Screening and Comparison

Standard library searching is done in two steps:

a) A screening search (pre-search) locates a set of similar spectra (*search_type* = NISTMS_SCREEN_SRCH) and stores results in NISTMS_HIT_LIST. The number of hits returned by this operation will depend on the input spectrum and search type. It is recommended that space be allocated for at least 6,000 spectra (*max_spec_locs* = 6000). A minimum value of 120 is required. The maximum value is 6,000 (defined as NISTMS_MAX_FPOS in NISTMS.H).

b) A spectral comparison search (*search_type* = NISTMS_COMPARE_SRCH) compares library spectra located by the screening search to the user spectrum and returns library spectra in order of their computed spectral similarity values. Name and structural information for each hit may be optionally returned for display in a "hit list". Also, if a NISTMS_CONSTRAINTS structure has been allocated, only hits that satisfy specified constraints will be retrieved.

While a library search is being conducted, communication with the calling program is possible if a *callback* function pointer is provided in the structure NISTMS_IO. Two message types are sent to the callback function in a NISTMS_CALLBACK structure. If the field WhatToDo is set to WRITE_MSGLINE, search progress is given in the field String (The last 5 bytes of String are reserved and should not be displayed). If WhatToDo equals TEST_CANCEL, you may terminate the search by setting field ReturnValue equal to 1. This will allow you to implement a CANCEL button to the user who wishes to terminate a search. Also, by including this function, you may enable your application to redraw its windows and react to user activities.

Two "match factors" are returned for each search, a "forward" match factor that uses all peaks and a "reverse" match factor that ignores peaks in the unknown spectrum not present in the library spectrum. Results may be sorted using either of these criteria (*impure* = 0 sorts on the forward values, inpure = 1 sorts on the reverse values). In addition, for Identity searches sorted by forward match factors, probabilities that each retrieval is correct are returned (in *hit_prob*[]) along with the relative likelihood that the correct match is in the library (*in_library_prob*). Details on the origin and meaning of these probability values is given in an article entitled "Estimating Probabilities of Correct Identification From Results of Mass Spectral Library Searches" in the *Journal of the American Society of Mass Spectrometry*, Volume 5, pages 316-323, 1994. Details of the performance of Identity search algorithms are presented in "Optimization and Testing of Mass Spectral Library Search Algorithms for Compound Identification" on pages 859-866 in the same volume of this journal.

VII-A. No Presearch Library Search

The no presearch EI library search (*search_type* = NISTMS_NO_PRE_SRCH) consists of a single step; it does not include screening (presearch filtering) stage. It compares the submitted (query) spectrum to all spectra present in the active libraries. Results in NISTMS_HIT_LIST are similar to those of NISTMS_COMPARE_SRCH with one exception: this search method may produce a hit list containing up to 400 spectra (that is, *max_spec_locs* = *max_hits_desired* = 400, defined as MAX_NOPRESRCH_HITS in NISTMS.H).

VII-B. MS/MS Library Search (NIST MS dll version 2.1.1 and later)

Similarly to EI library search, there are two kinds MS/MS search: (1) two steps, presearch and compare, and (2) single step, no presearch. The MS/MS search applies only to MS/MS spectra and MS/MS libraries that contain precursor m/z, either general MS/MS libraries or annotated libraries of peptide MS/MS spectra.

(1) Fast two step MS/MS library search: presearch (*search_type* = NISTMS_SCREEN_SRCH, *cntls.search_mode* = 'E', *cntls.precursor_ion_100mz* = 100 × precursor ion m/z), followed by the spectral comparison (*search_type* = NISTMS_COMPARE_SRCH). This search may retrieve only library spectra that have precursor ion m/z in a range of a given half-width centered on the precursor m/z. The MS/MS library must have files peak_pm0.dbu and peak_pm0.inu. This search is available in version 2.1.2.1 and later.

(2) Single-step no presearch MS/MS library search (*search_type* = NISTMS_NO_PRE_SRCH, *cntls.search_mode* = 'E'). There are two kinds of this search:

(2.1) "no presearch" (*cntls.precursor_ion_100mz* = 0) that is similar to "no presearch library search" described in the previous section;

(2.2) default presearch (*cntls.precursor_ion_100mz* = 100 × precursor ion m/z) which compares all library spectra that have precursor ion m/z in the range of a given half-width (a.k.a. precursor m/z tolerance) centered on the precursor m/z.

Peptide MS/MS library search provides specific characteristics of the reliability of the found spectra that are not provided by other spectrum searches.

At present, it is not possible for users to readily generate an annotated peptide MS/MS library for searching. This requires a series of processing steps and specific annotations that are still under development at NIST. Details of the annotation will likely change in future versions, so a details description is not given here.

MS/MS and Peptide-specific constraints (members of NISTMS_CONSTRAINTS, whose names start with *pep_*) are ignored for all types of library search except "no presearch" (Presearch OFF) MS/MS spectrum search and sequential search in MS/MS libraries containing peptide MS/MS spectra.

Important new features of MS/MS spectra are non-integer peak m/z and two additional decimal places in peak abundances. In addition, MS/MS libraries may contain annotations for each peak, which are used during peptide MS/MS search in peptide library to provide higher reliability of the results. This additional information together with precursor ion m/z is located in to the NISTMS_MASS_SPECTRUM data structure.

Due to non-integer nature of accurate product peak m/z and precursor m/z, MS/MS Library Search requires tolerance settings for these values.

Several search mode flags may be applied to set the appropriate type of MS/MS presearch and search. For details, see NISTMS_SRCH_CONTROLS section in the DATA STRUCTURES chapter.

NIST MS dll needs special one-time initiation to enable the MS/MS library search: *search_type* = NISTMS_SET_VERSION; the current version string is "2.1.5". This initiation is recommended by default (to disable it use "2.0" string).

Two values of *search_type*, NISTMS_MARK_LIBS and NISTMS_MARK_ALL_LIBS, are designed to find out whether a library is compatible with a particular type of a search. They are described in section **XVII**, Checking Library Type and Size.

VIII. Sequential Search

All spectra in the library specified in *active_libs*[0] may be retrieved in a search that retrieves spectra in order of their ID number (*search_type* = NISTMS_SEQ_ID_SRCH). This search employs the NISTMS_HIT_LIST structure, returning up to *max_spec_locs* spectra locations in the *spec_loc* field. The actual number of spectral locations is returned in *num_hits_found.* This search starts when an ID number is given as an ASCII string in *string_in* in the main NISTMS_IO structure. This retrieves the next *max_spec_locs* spectra starting with the smallest ID number equal to or greater than one specified. Subsequent calls retrieve spectra in order by setting *string_in* = NULL. When *num_hits* is less than *max_spec_locs* , the end of the library has been reached. A typical application of this search is to use it with constraints designed to select spectra of compounds that have specific features.

IX. Data Retrieval

Data associated with a spectrum may be obtained by entering its location, obtained from one of the above searches, in *input_spec_loc*, using *search_type*.= NISTMS_GET_SPECTRUM_SRCH. Certain types of data will be returned only if the corresponding buffer has been allocated (a NULL pointer to this buffer means do not return data). This applies to alternate names (synonyms), contributors or comments, replicate spectra locations, and chemical structures. Since each of these types of data is resident in a different file, retrieval requires additional disk access and data retrieval will be faster if the fields holding unnecessary data are set to NULL.

1. Main Spectral Data
Library spectra are returned in *libms* (type NISTMS_MASS_SPECTRUM) and additional data (primary chemical names, formula, etc.) are returned in *aux_data* (type NISTMS_AUX_DATA).

2. Synonyms
If the *synonyms* field in *aux_data* contains a pointer to a buffer whose length is passed in *synonyms_len*, alternate chemical names, if available, will be returned in that buffer. The number of names found is returned in *num_synonyms* and names will be separated by NULL characters.

2.1 Tagged Synonyms *
Currently there are 27 tags used in synonyms: $:01, $:02… $:27. A tagged synonym starts with this 4-character tag. From the user's viewpoint, these tags are "Spectrum_type", "Compound_type"… "Peptide_mods". A list of these tags and functions, which allow synonym tag recognition and conversion, may be found in files nist_syntag.h and nist_syntag.c included in CallDll C-language sample code, and in the class SYNONYM_TAG included in .NET wrapper files NISTMSCL.h and NISTMSCL.cpp. See also sections "Tags in Spectrum Text Information" and "List of acceptable special tags" in this document.

3. Contributors and Comments
If *contributor* points to an allocated buffer of length *contributor_len*, contributors (or comments in case of a user library) will be returned as a NULL-terminated string *contributor*. Setting *contributor* to NULL instructs the program to not return this data.

4. Replicate Spectra
Locations of replicate spectra, if available, may be obtained along with the principal data for a spectrum in the main NIST library (NISTMS_MAIN_LIB). These locations are returned in the allocated buffer *rep_locs* capable of holding up to *num_rep_locs* values (10 will be more than enough). The number of locations actually returned is given in *num_rep_locs*. Retrieved locations may be used with the NISTMS_SPECTRA_SRCH to retrieve data by passing the appropriate *rep_locs* value in *input_spec_loc* (in the NISTMS_IO structure).

5. Chemical Structures
If the chemical structure pointer (*stdata* in NISTMS_IO) is non-NULL and structural information is available, the chemical structure will be returned in *stata*.
Also, structure locations obtained in an earlier search (*stru_pos* in NISTMS_AUX_DATA or *stru_pos*[i] in NISTMS_HIT_LIST) may be used to obtain structures using *search_type* = NISTMS_STRU_SRCH with the structure location given in *input_spec_loc* (NISTMS_IO). In this case a non-zero *molfile_handle* in NISTMS_IO means the structure should be retrieved from the user-supplied structure file previously opened with *search_type* = NISTMS_OPEN_MOLFILE (see next section, Handling Structures)

IX-A. Handling Chemical Structures

User-provided chemical structures may be attached to spectra for display and incorporation into libraries[8]. These structures may be in one of two documented formats: 1) a MOLfile containing a single structure, and, 2) an SDfile containing any number of structures (see A. Dalby, J. G. Nourse et al, J. Chem. Inf. Comput. Sci., 1992, *32*, 244-255; only generic [**G**] features are recognized by the software). If a name is not given with a structure (the name is in the first line in each structure), the name will be created from the file name and the structure position in the file.

A user-supplied MOLfile or SDfile may be opened using *search_type* = NISTMS_OPEN_MOLFILE, with *string_in* pointing to the full pathname of the file. The result is a negative value in *molfile_handle*. This value should be used in *search_type* = NISTMS_CLOSE_MOLFILE after operations on the file are completed. Up to NISTMS_MAX_USER_STRUCT_FILES = 4 files can be open concurrently, but only one at a time may be used in a search.

Below are all other *search_types* for handling structural information.

search_type = NISTMS_SCAN_USER_STRU_SRCH scans structural parts of user libraries or an opened MOLfile or SDfile. The *string_in* points to the sequence number (in ASCII form) of the first structure to be examined; the source of structures are either that referred to by a non-zero *molfile_handle* or the single user library pointed by *active_libs*[0]. The rest of structure information is contained in the NISTMS_USER_STRUCT_INFO structure pointed to by *user_struct_info* member of NISTMS_IO. The following is a description of NISTMS_USER_STRUCT_INFO members.

a) Information about up to *max_structs_desired* structures will be returned in allocated buffers:
b) *struct_names* (allocated length ≥ *max_structs_desired*max_one_struct_name_len*) must be large enough to receive up to
c) *max_structs_desired* NULL-terminated names of structures, up to *max_one_struct_name_len* bytes each;
d) *stru_pos* points to the buffer for file offset values for each structure (these values can be used for *search_type* = NISTMS_STRU_SRCH)
e) *stru_seq_nums* points to the buffer for the sequence numbers of the structures.
f) To scan only the structures whose names contain a search string, *name_filter* should point to that string.
g) After the scan is completed, *num_structs_found* contains the number of found structures.

search_type = NISTMS_STRU_SRCH reads a chemical structure from an open MOL/SDfile or MS library into the buffer pointed by *stdata* member of NISTMS_IO. Any combination of the following four types of data can be used to retrieve the structure: (1) *input_spec_loc* or, if it is zero, then *aux_data->stru_loc* as a structure location; (2) *aux_data->spec_loc*; (3) *aux_data->ident* and *active_libs*[0]; (4) *aux_data->casno* or, if it is zero, then *aux_data->specno* (to search Main Library only). If both *aux_data->spec_loc* and *active_libs*[0] are specified, they should refer to the same library. The software attempts to find a chemical structure using the data in the following order:

[8] The easiest way to create a mass spectral library containing chemical structures is to create an SDfile containing both chemical structures and mass spectra and use Lib2NIST converter to create a library out of this file. Examples of SDfile format are files STRUSAM2.SDF and STRUSAMP.SDF installed with NIST/EPA/NIH MS Library.

If *molfile_handle* is not zero then only (1) will be used. Otherwise, for Main Library: (1), (3), (4), (2→4); for Replicates Library: (1), (4) , (2→4), (3→2→4); for User Library: (1), (2), (3→2), (4). Digits after arrow "→" specify what kind of data is retrieved using data of the type shown before the arrow.

A non-zero *molfile_handle* is used by all *search_type* values mentioned above in this section and by *search_type* = NISTMS_ADD_TO_LIBRARY_SRCH (see section "Maintaining User Libraries" below).

A one-time procedure for checking/indexing structural parts of user libraries must be called after the first DLL initiation; *search_type* = NISTMS_INDEX_USER_STRU; *active_libs* pointing to a list of all user libraries to be used (other kinds of libraries will be ignored by the call). This ensures the proper maintenance of user libraries.

search_type = NISTMS_MAKE_MOLFILE creates or appends a molfile by either (a) converting *stdata* into a molfile or (b) copying a molfile from the user library:

(a) *string_in* points to a string containing a
full molfile name or
"full molfile name" "molfile title[\rmolfile comment]"
where \r is carriage return symbol (0x0D), brackets enclose the optional string;
stdata contains previously read chemical structure
aux_data = NULL;

(b) *string_in* points to a string containing a full molfile name
stdata = NULL;
aux_data. stru_loc contains previously found location of a structure in a user library

search_type = NISTMS_ALT2AROM makes changes to the structure saved in *stdata*:
string_in points to a string containing a number made out of a bits described below
stdata contains a structure to which the changes are to be applied

Bit	Value	Action
0	0	Convert alternating single/double bonds to aromatic
0	1	Convert aromatic bonds to alternating single/double
1	2	Denormalize N-oxide: $N^{(IV)+}$— O^- => $N^{(V)}$=O
2	4	Remove all charges, radicals, carets from atom names
3	8	Remove charges, radicals, carets only; don't do arom<=>alt conversion or denormalization

Example: *string_in* = "6" means (since $6 = 0\times1+1\times2+1\times4 = 0\times2^0+1\times2^1+1\times2^2$)
Convert alternating single/double bonds to aromatic (0×1, bit 0 contributes 0)
Denormalize N-oxide (1×2, bit 1 contributes 2)
Remove all charges, radicals, carets from atom names (1×4, bit 2 contributes 4)

Note 1. To effectively use structure indexing it is recommended that different structures being added to the user library from the MOL/SDfile(s) have different names or, if names in the MOLfiles are not available, MOLfiles containing different structures should have different short (8.3) names.

Note 2. *stru_pos* contains a 4-bit library number and 28-bit structure file offset. If a user library structure file (USRSTRUC.DB, which has SDfile format) size exceeds 2^{28}-1 = 268,435,455 bytes then stru_pos cannot be used to retrieve or address a chemical structure. However, a structure associated with a mass

spectrum may be retrieved together with the spectrum without using stru_pos. This restriction is lifted in the new 2GB API, where *stru_pos* has NISTMS_RECLOC type.

X. Constraints and Hit List Utility

As a convenience for programmers, a set of *num_hits_found* spectra locations in *spec_locs* (both in *hit_list*) may be further processed. This call can reduce the set of spectra using any of a number of optional constraints and can also return chemical identification information of use for hit list presentation. The final set of spectra locations is returned in *spec_locs*, with the number of these in *num_hits_found*. All of this processing may be done without this utility by first retrieving data for each spectrum individually, then testing parameters and finally saving hit list information.

This utility is invoked with search_type = NISTMS_BUILD_HITLIST, a NISTMS_HIT_LIST structure from an earlier search and, optionally, a NISTMS_CONSTRAINTS structure holding user-specified constraints. This call is not needed for library-search results, since constraints and hit list information can be processed more efficiently along with spectral comparison (see Library Search, section **VII**).

XI. Maintaining User Libraries

Spectra contained in structure *usersp* along with identification information in *aux_data* are added to a user library specified by *active_libs*[0] using *search_type* = NISTMS_ADD_TO_LIBRARY_SRCH. Upon return *aux_data->ident* contains the library identification number (ID) assigned to the added spectrum. A chemical structure can only be added to the user library together with the spectrum. The location of the structure should be indicated with *stru_loc* in *aux_data.* The structure can be located either in any initiated user library or in the previously opened user-supplied MOLfile specified by non-zero *molfile_handle* in NISTMS_IO. To obtain *stru_loc* values of the structures use search_type = NISTMS_SCAN_USER_STRU_SRCH. *stru_loc* values pointing to the structures in the user libraries can also be obtained in searches retrieving spectra and hit lists; these structures will automatically be copied together with the spectrum to a user library. If *stru_loc* points to the structure located in the Main Library, that structure will not be added to the user library. To associate a structure from Main Library with the spectrum being added to the user library, *aux_data->casno* and *aux_data->specno* of the spectrum must be the same as in the Main Library spectrum associated with this structure. To associate a structure with a spectrum already in the user library four actions are needed: (1) read the spectrum from the library and save its ID, (2) set appropriate *aux_data->stru_loc* and, if necessary, set *molfile_handle* value to the NISTMS_IO structure referencing the spectrum, (3) add the spectrum back to the user library; (4) delete old spectrum from the library using previously saved ID. (Note that the spectrum is retrieved to the buffer pointed by *libms*, while the spectrum is added to the library from the buffer pointed by *userms*.)

To include alternate names (synonyms) to the user library record, *synonyms*, *synonyms_len*, and *num_synonyms* in *aux_data* should be set.

Spectra are deleted (one at a time) from the user library specified by *active_libs*[0] by providing their library identification numbers (*ident* in *aux_data*) in *string_in* as an ASCII string and performing calls with search_type = NISTMS_DELETE_FROM_LIBRARY_SRCH.

A creation of a new, empty user library is accomplished with the function CreateUserLibrary described in section XVI.

In order to maintain user library spectrum data self-consistent, the program replaces or calculates, if possible, the following NISTMS_AUX_DATA members:

NISTMS_AUX_DATA Member	Calculated from	Note
mw	NISTMS_AUX_DATA:: formula	If mw=0 and formula is valid
exact_mw	NISTMS_AUX_DATA:: formula	If formula is valid
instr_type	Synonym $:06	If this synonym exists

If a library spectrum does not fit in the largest record length allowed, synonyms, starting from the last one, are removed.

XII. Molecular Weight Estimation

This search calculates molecular weight (MW) of compound based on hit list of spectra retrieved in a library search. Estimating MW is done in two steps:
a) Library search (A screening search (*search_type* = NISTMS_SCREEN_SRCH) and a spectral comparison search (*search_type* = NISTMS_COMPARE_SRCH) (see Library Search, section **VII**)
b) MW estimation search (*search_type* = NISTMS_MW_EST).

The search can estimate up to 5 possible MW values for the given compound. Calculated MW values and their probabilities are returned in arrays *interp_ms.mw_est[ ]* and *interp_ms.mw_est_prob[ ]* respectively.

XIII. Molecular Weight Estimation 2

This search has been excluded from the DLL starting from 1999 release.

XIV. Chlorine - Bromine Estimation.

This search provides information about presence/absence Chlorine - Bromine in unknown compound. The search returns its results in the following fields in **INTERP_MS** structure:

cl is the predicted number of Cl atoms
br is the predicted number of Br atoms
clbr_prob is the probability that cl and br above are correct
prob_any_clbr is the probability that compound has Cl or Br
any_cl_prob is the probability that compound has Cl
any_br_prob is the probability that compound has Br.

Also, Chlorine - Bromine Estimation Search can output warning messages. The messages are located in the following fields of **INTERP_MS** structure

num_clbr_warnings is the number of the messages
clbr_warning[ ] is the buffer of the messages.

XV. Substructure Information.

This search provides information about presence/absence of structures in the unknown compound. The text file ”subtypes.dat” is an ASCII file of all possible substructures and their index numbers.

Substructure Information search is done in two steps:

a) Library search (A screening search (*search_type* = NISTMS_SCREEN_SRCH, *cntls.search_mode* = 'S', Similarity Search is recommended) and a spectral comparison search (*search_type* = NISTMS_COMPARE_SRCH; see Library Search, section **VII**)

b) Substructure Information search (*search_type* = NISTMS_SUBSTR_SRCH).
Input for the search is a previously found set of similarity numbers and CAS registry numbers (*hit_list->sim_num[ ]*, *hit_list-> casnos[ ]*, and *hit_list->num_hits_found*).
The search returns its results in array *substru_prob[ ]* of **INTERP_MS** structure. The array contains probabilities for the presence of structures listed in “subtypes.dat” file. If a probability value is negative, the absolute value is the probability of substructure absence; if positive, it is the probability that the substructure is present. The field *num_substrus* of **INTERP_MS** structure contains the number of substructures processed.

XVI. Creating New User Libraries

Creation of a new, empty user library is accomplished with the function:

int CreateUserLibrary(char *** NISTMS_path,** char * **new_lib_path**).

The parameters are zero delimited strings containing directory paths. Note that for the NIST MS Search Program, each library is represented by a directory under the main program directory (MAINLIB and REPLIB, for example, are the predefined names of the NIST/EPA/NIH main and replicates libraries). Non-zero returned values signify an error (see NISTERR.H).

If the directory that will contain the new user library does not exist, it will be created. If the target directory already exists and contains a minimal set of user library files, the function will fail.

NISTMS_path provides the full path to the directory containing the NIST search software (MAINLIB is a subdirectory of this directory). This is needed to link the new library to the NIST MS search program.

The string **new_lib_path** specifies the directory where the new library will be stored. There are two ways of specifying the location of a new user library.

1) *In any user-specified directory* (indirect). In this case **new_lib_path** is the full path to a new or pre-existing empty directory. In this case, data files are stored at **new_lib_path** and a directory with the same name created under the directory **NISTMS_path**. In the new directory under **NISTMS_path** a file **alias.msd** is created that contains **new_lib_path**.
2) *In the default location* (direct). In this case a simple directory (library) name is given in **new_lib_path** and a user library with this name is created under main program directory (as given in **NISTMS_path**).

Note that a new user library MUST have a name different than any existing library.

Deletion of a user library is simply accomplished by deleting the appropriate subdirectory of the main program directory after closing all library files with search_type=NISTMS_CLOSE_SRCH.

XVII. Checking Library Type and Size

A. GetLibraryType Function. The function **GetLibraryType (char* pLibPath, int bufferSize)** returns the type of library referred to in the pLibPath directory. The buffer **pLibPath** is zero delimited string containing the path of the directory containing library. If the library is "aliased" (only referred to from the **pLibPath** directory with an *alias.msd* file), the actual library directory is returned in pLibPath (in this case you must allocate a sufficiently large buffer for **pLibPath** and provide its length in **bufferSize**).

The function returns the following values

Return Values		Directory contains
NISTMS_MAIN_LIB	1	NISTMS Main Library
NISTMS_USER_LIB	2	User Library
NISTMS_REP_LIB	4	NISTMS Replicate Library
NISTMS_NOT_A_LIBRARY	-1	Not a library

B. GetNumberOfEntries Function. The function long **GetNumberOfEntries(char * pLibPath)** returns the number of spectra in a library. Negative returned values indicate and error:

Possible Errors Codes		Description
ERR_CANNOT_INIT_CTREEGV	-3	Cannot start CTREE function
ERR_CANNOT_INIT_CTREE	-4	Cannot initialize CTREE
ERR_BAD_INPUT_DATA	-102	Cannot find or access library

C. Determining Search Types Available for a Library. EI libraries may not be searched with the MS/MS type of search: the search would always return no hits from an EI library. For historic reasons, some MS/MS libraries may not be compatible with certain types of MS/MS search[9]. Some libraries may not be searched for NIST r.n., Any Peaks Exact Mass, etc. The following two searches provide information on whether a certain search is compatible with a library or the search would ignore a library and always return no hits from it.

C1. Input:
search_type = NISTMS_MARK_LIBS
string_in points to a string containing a library test type, a single character from "1" to "6"
active_libs contains a list of initiated libraries to be checked on compatibility with the Search Type
aux_data points to an existing AUX_DATA structure
Output:

[9] Rebuilding such a MS/MS library with Lib2NIST may make it suitable for all types of MS/MS search.

aux_data.name string contains as many characters `' '` (space) and `'!'` (exclamation) as the number of **active** libs. A space means the library is compatible with the desirable type of the search; the exclamation means it is not compatible.

C2. Input:
search_type = NISTMS_MARK_ALL_LIBS
string_in points to a string containing a library test type, a single character from `"1"` to `"5"`
active_libs **is ignored**
aux_data points to an existing AUX_DATA structure
Output:
aux_data.name string contains as many characters `' '` (space) and `'!'` (exclamation) as the number of **initiated** libs. A space means the library is compatible with the desirable type of the search; the exclamation means it is not compatible.

test type 1: MS/MS search
search_type = NISTMS_SCREEN_SRCH (2-step search)
cntls.precursor_ion_100mz > 100
cntls.search_mode='E'
test type 2: MS/MS search
search_type = NISTMS_NO_PRE_SRCH (1-step search)
cntls.precursor_ion_100mz > 100
cntls.search_mode='E' or 'F'
test type 3: NIST r.n. search
search_type = NISTMS_NISTNO_SRCH
test type 4: Any Peaks Search Exact Mass
search_type = NISTMS_ANYPEAK_ONE_PEAK_SRCH
peak_info.type = NISTMS_EXACT_MASS_PEAK
test type 5: CAS r.n. search including Related and/or Salt/Mix CAS r.n.
search_type = NISTMS_CASNO_SRCH2
Note: if Related and/or Salt/Mix CAS r.n. are not indexed in a library, the search automatically falls back on NISTMS_CASNO_SRCH for this library.
test type 6: Exact Molecular Mass search
search_type = NISTMS_EXACT_MASS_SRCH
Note: A library may be (re)indexed for this search with *search_type* = NISTMS_INDEX_LIBRARY_EXACT_MASS.

D. Library Files Responsible for Various Search Types. The following provides tentative information on whether a library is a MS/MS library or has some other features. Note that EI spectra may be included in a MS/MS library.

MS/MS libraries may have the following files, which usually are not present in EI libraries:

File name(s)	Lib.Type	Description
msztxt.dbu msztxt.inu	MS/MS	m/z, abundance, and (optionally) peak annotations; indexed by library spectrum ID.
precmz.inu	MS/MS	Index of precursor m/z location in user.dbu
mzbin.dbu mzbin.inu	MS/MS	Minimal spectrum information for faster spectrum comparison in unconstrained MS/MS search; indexed by spectrum ID.
precmzb.inu	MS/MS	Index of precursor m/z location in mzbin.dbu
mzbinpr.inu	MS/MS	Index of protein name location in mzbin.dbu (peptide MS/MS libraries)
peak_pm0.inu peak_pm0.dbu	MS/MS	Presearch index and peak lists for the MS/MS search; needed for the two-step MS/MS search
references.txt	MS/MS	Contains spectrum references (source information)
peak_em.dbu peak_em.inu	any	Any peaks search for Exact Mass peaks index files
registry2.inu	any	Index for CAS r.n. search including Related and/or Salt/Mix CAS r.n.
specno.inu	any	NIST r.n. search index file
exactmw.inu	any	Exact Molecular Mass search index file

The default MS/MS presearch (one-step search) requires either precmz.inu or precmzb.inu, mzbin.dbu, and mzbin.inu files; the two-step MS/MS search requires peak_pm0.inu and peak_pm0.dbu.

DATA STRUCTURES

Data transfer between the DLL and the calling program is done through data structures defined in NISTMS.H. Memory allocation of these structures and various variable length buffers within them is the responsibility of the calling program. References to defined structures appear only in the parent data structure, NISTMS_IO.
Note. If an order of a data structure members described in this document differs from that in the NISTMS.H, or if you discover other discrepancies between NISTMS.H and NISTDLL.DOC, use information from the NISTMS.H

NISTMS_IO			
	Type	**Field Name**	**Description**
For libraries initiation only			
1	unsigned	**num_libs**	number of libraries available
2	char *	**lib_path**	paths to available libraries
3	char *	**lib_types**	libraries types
4	char *	**work_dir_path**	path to work directory
5	void (* callback)(IQ*)	**callback**	message passing
Input information only			
6	char *	**active_libs**	libraries to be accessed in a search
7	NISTMS_RECLOC	**input_spec_loc**	spectrum or chemical structure location
8	char*	**string_in**	information needed for certain searches
9	NISTMS_MASS_SPECTRUM*	**userms**	input user mass spectrum
10	NISTMS_SRCH_CONTROLS*	**cntls**	information for library searching
11	NISTMS_CONSTRAINTS*	**constraints**	optional constraints
Output information only			
12	NISTMS_RECLOC	**output_spec_loc**	spectrum location
13	NISTMS_MASS_SPECTRUM*	**libms**	mass spectrum retrieved from a library
14	NISTMS_STDATA *	**stdata**	chemical structure data
15	int	**error_code**	Error codes returned from DLL
May contain input and output information			
16	NISTMS_HIT_LIST*	**hit_list**	list of retrievals
17	NISTMS_INC_NAME_INFO*	**name_info**	data for incremental name searching.
18	NISTMS_PEAK_INFO*	**peak_info**	data for "anypeak" searches
19	NISTMS_AUX_DATA*	**aux_data**	non-spectral information
20	INTERP_MS*	**interp_ms**	data for substructure identification
21	NISTMS_USER_STRUCT_INFO*	**user_struct_info**	user structures scanning data
22	int	**molfile_handle**	handle of an open MOL/SDfile or 0

1) num_libs is the number of libraries available.

2) **lib_path** is a pointer to a null-terminated string containing DOS paths to available libraries. The number of paths required is num_libs, with each path separated from others by '\r' (carriage return").

3) **lib_type** is a null-terminated string (length = **num_libs**) containing library types. The i-th character corresponds to the i-th library in **lib_path**. Types of libraries are:

NISTMS_MAIN_LIB	the NIST/EPA/NIH main library
NISTMS_REP_LIB	the NIST/EPA/NIH replicate spectra library
NISTMS_USER_LIB	user-created or other library

4) **work_dir** is a null-terminated string containing path to the work directory.

5) **void (* callback)(IQ*). callback** is a pointer to an external function that you provide for communication with the search program during a library search. It takes a pointer to a NISTMS_CALLBACK structure as its argument (allocated by the DLL). Search progress is reported in the String field when WhatToDo is equal to WRITE_MSGLINE. When WhatToDo is equal to TEST_CANCEL, you may abort the search process by setting ReturnValue equal to 1. Registering this function may be done only during program initialization (*search_type* = NISTMS_INIT_SRCH). The last 5 bytes of String are reserved and should not be displayed.

6) **active_libs** is a null-terminated string specifying the libraries to be accessed in a search. A library is specified by its sequence number in **lib_path** and **lib_type** (the first library is 1, the second is 2 and so on). Some searches use only the first active library **active_libs**[0] or do not use **active_libs** at all.

7) **NISTMS_RECLOC input_spec_loc** is a spectrum or chemical structure location retrieved in an earlier search. In dll versions previous to 2GB API (ver. 2.1.3.1), NISTMS_RECLOC is a 32-bit integer. In 2GB API, NISTMS_RECLOC has size 6 bytes:

NISTMS_RECLOC (2 GB API only)		
Type	**Field Name**	**Description**
long	**loc**	Library datafile offset (> 0)
char	**ftype**	Type of the data or index file (≥ 0)
char	**lib**	Library number (≥ 0)

8) **string_in** points to an ASCII null-terminated string containing information needed for certain searches.

9) **NISTMS_MASS_SPECTRUM * userms** contains a mass spectrum (the number of peaks with two arrays of integers, one for masses and one for abundances). **userms** points to a user-supplied spectrum for library searching and building. The following table describes the data structure fields

NISTMS_MASS_SPECTRUM		
Type	**Field Name**	**Description**
unsigned int	**num_peaks**	Number of mass spectral peak
unsigned int[*NISTMS_MAXPEAKS*]	**mass**	Array of m/z values
unsigned int[*NISTMS_MAXPEAKS*]	**abund**	Array of abundance values (999 max)
long	**precursor_ion_100mz**	precursor ion m/z × 100 or precursor ion m/z in f32 format
int	**num_exact_mz**[1]	number of "exact" MS peaks
char **	**exact_mz**[1]	array of pointers to "exact" MS peaks
int	**exact_mz_len**[1]	number of pointers allocated
char *	**buf_exact_mz**[1]	buffer for "exact" MS peaks
int	**buf_exact_mz_len**[1]	length of the allocated buffer

[1] Each pointer *exact_mz*[i] (i = 0.. *num_exact_mz*-1; *num_exact_mz* ≤ *exact_mz_len*) points to an "exact" MS peak located in *buf_exact_mz*. "Exact" MS peak is a NULL-terminated string containing text `<m/z>t<abundance>t<annotation>` where "`t`" is a TAB character. If the `<annotation>` is not available then the preceding it TAB is not present. Unlike .MSP file, the annotation in NISTMS_MASS_SPECTRUM must NOT be enclosed in double quotes. Annotations that begin with letters `?,a,b,c,x,y,z,p,I,` or 5 letters `iTRAQ` are reserved for NIST peptide libraries.
The number of "exact" peaks, *num_exact_mz*, may exceed 800 and reach 7,000-8,000. The limit is determined by constants *NISTMS_DFLT_MAX_PEAK_TXTDATA_LEN*=140000 and *NISTMS_DFLT_MAX_PEAK_TXTDATA_NUM*=10000 defined in NISTMS.H. "Exact" MS peaks should have different m/z. Max. abundance should be 999.
Only MS/MS spectra that have parent ion m/z may have "exact" MS peaks.
To fill out *mass, abund, and num_peaks* from *num_exact_mz, exact_mz, and buf_exact_mz* containing a mass spectrum, one may use introduced in version 2.1.5 *search_type*= NISTMS_EXACT_MZ_TO_INT_PEAKS. Note that abundances of "Exact" MS peaks that have the same integer *mass* value are summed. In previous to 2.1.5 version of CallDll.c example the largest value was erroneously chosen for the integer abundance.

NISTMS_MAXPEAKS is defined in NISTMS.H as **800**. Peaks (**mass**[i], **abund**[i]) should be in order of increasing mass (m/z), and abundances should be normalized to 999. No two peaks should have the same mass.

10) NISTMS_SRCH_CONTROLS contains information needed to conduct library searching with a user-provided spectrum (given in **userms**). The following table describes the data structure fields

Search mode Q and **I** are quick and normal identity searches, **P** is like **I** except that penalties are applied to rare compounds, **S**, **L**, and **H** are similarity searches using absolute (conventional) peaks only, neutral loss only and both (hybrid), respectively. **M** is similar to **S**, however, it uses a different scheme for calculating match factors and penalizes spectra that do not have a peak with m/z=MW of the precursor. **E** – MS/MS search – is a kind of an identity search; its match factors are different from all other searches and, in case of NIST Peptide MS/MS libraries, take into account peak annotations. Usually, MS/MS search requires precursor ion m/z, precursor ion m/z tolerance, and

product ion m/z tolerance to be set in NISTMS_SRCH_CONTROLS data structure[10]. Searches **L**, **H**, and **M** require molecular weight of the search spectrum, *user_mw*.

NISTMS_SRCH_CONTROLS		
Type	**Field Name**	**Description**
int	**search_mode**[3]	'**Q**'uick, '**I**'dentity (normal or **P**enalized), '**S**'imilarity, '**L**'oss only, '**H**'ybrid, '**M**'S/MS in EI library, or '**E**' (MS/MS) search
int	**user_mw**	for neutral loss searching (required with '**L**' or '**H**' search modes)
int	**impure**	= 0, uses all peaks, = 1 uses only peaks present in library spectrum
int	**min_mass**	lowest mass for spectral comparison; 0 is same as 1, -1 means greatest out of the lowest masses of the library spectrum and the **userms**; see discussion below
int	**max_mass**	highest mass for spectral comparison; 0 is same as 2000, -1 means greatest out of the library spectrum and the **userms** masses (no upper limit).
int	**min_abund**	lowest abundance used (base peak = 999)
float	**precursor_ion_tolerance**[1,6]	m/z half-interval multiplied by 100, used in presearch
float	**product_ions_tolerance**[1,6]	m/z half-interval multiplied by 100, used in matching
unsigned	**precursor_ion_100mz**[1,7]	precursor m/z multiplied by 100, or precursor m/z in f32 format or 0 for peptide no presearch search
char	**pep_bTF_qry**[2]	0 or 1, output flag for T/F-qry score
char	**pep_bE_Omssa**[2]	0 or 1, output flag for E-Omssa score
char	**pep_bTF_lib**[2]	0 or 1, output flag for T/F-lib score
char	**bRevImpure**[1,4]	0 or 1, output flag for Rev-Dot (reverse dot product)
char	**pep_bOmssa**[2]	0 or 1, weighing flag for spectra comparison
char	**pep_bNumReplicates**[2]	0 or 1, weighing flag for probability calculation
char	**pep_bQ_TOF**[2]	0 or 1, weighing flag for spectra comparison
char	**pep_cThreshold**[1]	2=low, 1=medium, 0=high; used in all MS/MS searches
int	**pep_nCysteineModification**	unused
char	**pep_bRefPeakFraction**[5]	0 or 1, output flag for percent of matched reference peaks

[1] Meaningful only in MS/MS library search.
[2] Meaningful only in peptide MS/MS library search in annotated NIST peptide libraries.
[3] Also see Search mode flags section below.

[10] NIST MS dll versions prior to ver. 2.1.2.1 accepted MS/MS *search_type* = NISTMS_NO_PRE_SRCH only; version 2.1.2.1 and later also accept *search_type* = NISTMS_SCREEN_SRCH followed by *search_type* = NISTMS_COMPARE_SPECTRA_SRCH for faster MS/MS search. Starting from version 2.1.3.1, various types of MS/MS libraries may be searched in a single search. However, if a library is not compatible with a search type, no hits will be returned from such a library. Use *search_type* = NISTMS_MARK_LIBS to determine the compatibility.
Search mode **F** briefly introduced in earlier versions as a kind of MS/MS search is obsolete and replaced with **E**.

[4] Was missing from documentation from 05/15/2008; first time documented in version 2.1.5.3
[5] Available in version 2.1.5.3
[6] In versions previous to 2.1.5.3 these values have type int
[7] In versions previous to 2.1.5.3 this value has type long

Search mode flags (available in version 2.1.2.1 and later) together with the *search_mode* letter and *search_type* define the way the spectrum is searched in a library. To set search mode flags, replace the *search_mode* = X (X=**Q**, **I**, **P**, **S**, **L**, **H**, **M**, **E**) letter's ASCII code with a result of its bitwise OR with one or more of the search flags: *search_mode* = ('X' | search_mode_flag).

Search mode flag	Description
SEARCH_MODE_FLAG_FAST_PRESEARCH	When *search_type*=NISTMS_SCREEN_SRCH (two-step search), perform faster search by reducing the number of spectra selected for the 2nd step, the spectra comparison
SEARCH_MODE_FLAG_IGNORE_PRECURSOR	In MS/MS search, when *precursor_ion_100mz* ≥ 100, ignore peaks within m/z interval ±*precursor_ion_tolerance*/100 centered on m/z=*precursor_ion_100mz*/100
SEARCH_MODE_FLAG_ALT_PEAK_MATCHING[1]	In MS/MS search use alternative peak matching when comparing two spectra
SEARCH_MODE_FLAG_GENERIC_MSMS[1]	Generic (non-peptide) MS/MS search, which produces a different score and does not use peak weighting and annotations in ms/ms spectra compare
SEARCH_MODE_FLAG_REJECT_OTHER[1]	Reject peptide library spectra in Generic and non-peptide library spectra in Peptide MS/MS spectra compare
SEARCH_MODE_FLAG_PRECUR_MZ_TOL_PPM[1]	Precursor m/z tolerance is in ppm instead of 100*(delta m/z)
SEARCH_MODE_FLAG_PROD_PEAK_TOL_PPM[1]	Product peak m/z tolerance is in ppm instead of 100*(delta m/z)

[1] Available in version 2.1.5

A special *min_mass* option in all kinds of user spectrum searches (except **L**oss and **H**ybrid search) is available: "min mass not greater than X".

To turn it on, set *min_mass* < -1 (negative), which makes X = abs(*min_mass*).
As the result, all peaks with m/z >= abs(*min_mass*) shall always be compared. That is, the comparison starts at the median (middle) of three values: abs(*min_mass*), the lowest mass of the library spectrum, and the lowest mass of the search spectrum.

Presearch in a two-step search uses abs(*min_mass*) as a minimum mass.
If *min_mass* ≥ -1, the comparison works in a usual way: the comparison starts either at min_mass>0 or at m/z=1 if min_mass=0, or at the greatest out of the lowest masses of the library spectrum and the user spectrum if *min_mass* = -1. (Backward compatibility note: previously *min_mass* < -1 was treated the same way as *min_mass* = -1 is treated now.)

NIST MS Search Limits Minimum m/z vs. cntls->min_mass		
Minimum m/z (NIST MS Search)	**min_mass value**	**Spectra comparison starts at**
Off	-1	the greatest out of the lowest masses of the library spectrum and the search spectrum
equals to X	X	m/z=X (X>0; X=0 is the same as X=1)
never greater than X	-X	m/z = median(X, the lowest peak m/z of the library spectrum, the lowest peak m/z of the search spectrum); X>1

NIST MS Search Limits Maximum m/z vs. cntls->max_mass		
Maximum m/z (NIST MS Search)	**max_mass value**	**Spectra comparison ends at**
Off	-1	the greatest out of the highest peak m/z of the library spectrum and the search spectrum
On	X	m/z=X (X>0; X=0 is the same as X=1)

If the MS Search Apply Limits check box is unselected then Minimum m/z and Maximum m/z are set to Off.
In case of EI spectra search, the highest peak m/z compared is 2000. In case of MS/MS search (search_mode='E'), the highest peak m/z compared is 4000. In all cases, only peaks with m/z ≤ 2000 are indexed for the two-step presearch.

min_mass meaning in the Loss search and in the loss part of the Hybrid search.

The length = MaxLoss of the compare m/z range is defined from two search control parameters: *min_mass* and *user_mw* values in the following way:

(a) *min_mass* = -1 or abs(*min_mass*) ≥ *user_mw*:
MaxLoss = 64 (that is, comparison is made from highest m/z down to mw-64 value.

(b) *min_mass* = 0:
MaxLoss = user_mw (comparison uses the entire search spectrum)

(c) abs(*min_mass*) < *user_mw*:
MaxLoss = *user_mw* - abs(*min_mass*), that is, compare user spectrum from highest m/z down to abs(*min_mass*))

In other words, the neutral losses are compared up to MaxLoss mass units.

Note 1. When MaxLoss > 64 it is best to employ the no presearch option since the loss presearch peak index only includes loss peaks below 64 m/z.
Note 2. If abs(*min_mass*) ≥ *user_mw* is used with the "Loss" presearch then no spectra will be found.
Note 3. In the "Simple" part of the Hybrid presearch, if *min_mass* ≠ -1 then abs(*min_mass*) is treated as min. mass limit.
Note 4. For backward compatibility, use *min_mass* = -1 with the Loss Search.

11) NISTMS_CONSTRAINTS contains optional constraints that may be used to limit the hits returned to those having selected characteristics. It may be used along with two functions calls, NISTMS_COMPARE_SPECTRA_SRCH and NISTMS_BUILD_HITLIST. The following table describes the data structure fields

NISTMS_CONSTRAINTS			
Type	**Field Name**	**Description**	
unsigned int	**mw_min**[3]	minimum MW allowed	
unsigned int	**mw_max**[3]	maximum MW allowed	
char[40]	**name_frag**	name fragment; See "Syntax of Name Fragments Constraint" section in Appendix for details.	
int	**mode_el_list**	*NISTMS_EXACT* or *NISTMS_ELS_IN_LIST*	
int	**num_el_list**	number of elements specified, 0 means ignore constraint	
char[10][2]	**el_list**	list of 1 or 2 character element symbols	
int	**num_atom_comp**	number of elements specified for this constraint, 0 means ignore constraint	
char[10][2]	**comp_list**	element name	
char[10]	**sign**	one of these: < = >	
int[10]	**num_each_el**	number of elements' atoms	
int	**mode_peaks**	*NISTMS_REL_PEAKS* or *NISTMS_ABS_PEAKS*	
int	**num_other_peaks**	number of peaks specified	
unsigned int[10]	**abmin**	minimum abundance (% base peak) or rank	
unsigned int[10]	**abmax**	maximum abundance (% base peak) or rank	
unsigned int[10]	**mass**	mass of peak	
char[10]	**peak_type**	type of peak: 'N' (normal), 'M' (maxmass), 'L' (neutral loss) or 'R' (rank) (see section **VI,** Spectra With Specified Peaks)	
int	**max_misses**	not used	
int	**other_dbs**	Bit field with flags for presence in other databases	
char [1024]	**comment_tags**[2]	string(s) to match (possibly tagged) information in Comments. See "Syntax of Tags in Comment and Text Info constraint" section in Appendix for details.	
char [128]	**pep_name_frag**[1]	string(s) to match peptide sequence; See "Syntax of Peptide Sequence Constraint" section in Appendix for details.	
int	**pep_min_charge**[1,3]	minimum peptide charge	
int	**pep_max_charge**[1,3]	maximum peptide charge	
int	**pep_min_protons**[1,3]	minimum number of peptide protons	
int	**pep_max_protons**[1,3]	maximum number of peptide protons	
int	**pep_min_residues**[1,3]	minimum number of residues in a peptide	
int	**pep_max_residues**[1,3]	maximum number of residues in a peptide	
int	**num_top_isotopes**[4]	zero=monoisotopic mass only, otherwise, search among this number of most abundant peaks in the isotopic envelope	
unsigned int	**exact_mw_min**[4,5]	min. exact mass (f32)	To disable this constraint, set both values to zero
unsigned int	**exact_mw_max**[4,5]	max. exact mass (f32)	
unsigned int	**bits_instr_type**[4]	instrument type bitmap, see NISTMS_BIT_INSTR_TYPE, or 0 to disable this constraint	

[1] Used in peptide library with Peptide no presearch search and Sequential search only, available in version 2.1.1 and later

[2] Available in version 2.1.1, applies to all searches where constraints may be used

[3] To disable this constraint set the value equal to *NO_VALUE* defined in NISTMS.H

[4] Available in version 2.1.5

[5] Convert obtained elsewhere exact mass limits in Da to f32 format or use *search_type=* NISTMS_GET_EXACT_MASS_LIMITS to calculate *exact_mw_min* and *exact_mw_max* as members of NISTMS_PEAK_INFO::ExactPeak, then copy them to NISTMS_CONSTRAINTS.

NISTMS_EXACT means that the elements contained in each hit must be exactly those given in **el_list**. *NISTMS_ELS_IN_LIST* means that each element in a hit must be given in **el_list** (elements in hits may represent a subset of those in **el_list**).

Numbers of specific elements may be specified. The element i is in **comp_list**[i], the symbol greater than (>), less than (<), or equal to (=) is in **sign**[i] and the number in **num_each_el[i]**.

Peak specifications may be either given as independent abundance ranges (*NISTMS_ABS_PEAKS*) or relative to the first peak (*NISTMS_REL_PEAKS*). Peak types are those discussed for the *ANYPEAK* search mode. Abundances of peaks in *NISTMS_ABS_PEAKS* and the first peak the *NISTMS_REL_PEAKS* mode are percentages of the largest peak in the spectrum (the "base" peak). Abundances of other peaks in *NISTMS_REL_PEAKS* are relative to the first peak, hence may be greater than 100.

12) output_spec_loc is assigned a spectrum location in searches that can at most find one spectrum.

13) libms This holds a mass spectrum retrieved from a library (see **userms** above).

14) Chemical structure data are provided in a **NISTMS_STDATA** structure:

NISTMS_STDATA		
Type	**Field Name**	**Description**
int[NISTMS_MAXBONDS]	**xl** **yl**	x and y display coordinates, *i*-th bond is between **xl[2i]**, **yl[2i]** and **xl[2i+1]**,**yl[2i+1]**
int	**num_line_points**	internal use only
int	**num_points**	number of points given as **xl**, **yl** pairs: equal to number of bonds/2
char [NISTMS_MAXBONDS]	**bond_type**	single (1), double (2), triple (3), aromatic (4) (draw as single). **bond_type**[i] corresponds to bond between **xl[2i]**, **yl[2i]** and **xl[2i+1]**, **yl[2i+1]**
char[NISTMS_MAXCIRCS]	**xc** **yc**	center position of 10-pixel radius aromatic circle
int	**num_circs**	number of aromatic circles for display
int[NISTMS_MAXSTRINGS]	**xs** **ys**	coodinates of heteroatom strings, usually coincide with some of (**xl[i]**, **yl[i]**) pairs
int	**num_strs**	number of strings
int	**radpix**	internal use only
int[NISTMS_MAXSTRINGS]	**str_point**	internal use only
int[NISTMS_MAXSTRINGS] [NISTMS_MAXSTRINGLEN]	**str**	heteroatoms strings, **str[i]** contains heteroatom string **i**, located at **xs[i]**. **ys[i]**

This information is used for drawing a chemical structure. Procedures for displaying structures and extracting "connection tables" are available on request from NIST.

Stereo bond descriptors (Up, Down, Either) are extracted from the MOLfiles or User libraries structures.

Non-zero value of stereo = *io->stdata->bond_type*[i+*NISTMS_MAXBONDS*/2] describes a single bond (single bond *type io->stdata->bond_type*[i] = 0) connecting the first atom (located at the point {*io->stdata->xl*[2*i], *io->stdata->yl*[2*i]}) to the second atom {*io->stdata->xl*[2*i+1], *io->stdata->yl*[2*i+1]} for(i = 0; i < *stdata->num_line_points*/2; i++).

stereo > 0 means the wedge (pointed) end of the stereo bond is at the first atom; stereo < 0 means the wedge is at the second atom. abs(stereo) value means: 0 = not stereo, 1 = Up, 4 = Either, 6 = Down. Since stereo bond descriptors are located in the previously unused region of *io->stdata->bond_type*, they may be ignored for backward compatibility.

15) If a problem was encountered during processing, **error_code** will be set to a non-zero value. File NISTERR.H gives #defines for all possible **error_code**.values

16) **NISTMS_HIT_LIST** is a data structure holding a list of retrievals. Data locations are stored as is, optionally, information for showing the hits to the user.

NISTMS_HIT_LIST		
Type	**Field Name**	**Description**
NISTMS_SCREEN_SRCH uses only following three		
unsigned	**max_spec_locs**	maximum number of hits allowed, recommended 6000 = NISTMS_MAX_FPOS for library searches and not less than **num_hits_found** for filtering or building hit list
NISTMS_RECLOC*	**spec_locs**	allocated by caller, assigned values in DLL, length = **max_spec_locs** × sizeof(NISTMS_RECLOC)
int	**num_hits_found**	# spectra found and returned in **spec_locs**[] or initial number of spectra for filtering or building hit list
int	**max_hits_desired**	≤ **max_spec_locs**, largest number of hits returned in variable length buffers: **sim_num**, **rev_sim_num**, **stru_pos** , **lib_names, casnos, hit_prob.** Must be ≥ 120 for NISTMS_COMPARE_SPECTRA_SRCH.
Following fields used only for **NISTMS_COMPARE_SPECTRA_SRCH**		
int*	**sim_num**	similarity of library and user spectrum, all peaks used
int*	**rev_sim_num**	similarity of library and user spectrum, use only peaks present in library (optional, ignored if NULL)
The following two have meaning only for **Identity** search results sorted by forward match factors (impure=0)		
int*	**hit_prob**	probability (x100) that a retrieval is correct, assuming it is in the library (optional, ignored if NULL)
int	**in_library_prob**	relative likelihood that a matching compound has been found; has no meaning in case of MS/MS search
The following fields are optional for retrieving names and/or structures, if NULL or 0 they will be ignored		
char*	**lib_names**	buffer to receive a series of NULL terminated names
unsigned	**lib_names_len**	allocated length of **lib_names,** INPUT value
int	**max_one_lib_name_len**	maximum length of a name in **lib_names**, INPUT value
NISTMS_RECLOC*	**stru_pos**	file offsets to chemical structure, allocated length = **max_spec_locs** × sizeof(NISTMS_RECLOC) Values may be used later with NISTMS_STRU_SRCH
long*	**casnos**	positive values retrieved from NIST library are CAS registry numbers (*casno* in AUX_DATA); negative are NIST accession numbers (*specno* in AUX_DATA). The latter is provided only in case the former is zero. allocated length = **max_spec_locs** x sizeof(long) Values may be used later with NISTMS_STRU_SRCH
The following fields are optional for retrieving peptide library search specific scores		
float *[4]	**pep_Mf** [1]	peptide MS/MS library search specific match scores; allocated length for each of the pointers = **max_spec_locs** × sizeof(float). These scores are produced depending on **pep_bTF_qry, pep_bE_Omssa,** and **pep_bTF_lib** members of NISTMS_SRCH_CONTROLS; OUTPUT
char [4][12]	**pep_hdr_Mf** [1]	text headings for the peptide MS/MS library search specific match scores; if not available then the first byte in the 12-byte string is zero; OUTPUT

[1] Used in peptide MS/MS library search, available in version 2.1.1 or later.

Library names are returned in **lib_names** in adjacent buffers of length **max_one_lib_name_len** each. That is, name **i** (**i** ≥ 0) begins at location **lib_names[max_one_lib_name_len * i]**. The number of names actually returned is passed in **num_hits_found**, which is equal or less than **max_hits_desired**

17) NISTMS_INC_NAME_INFO is used for interactive incremental name searching. **num_names_desired** sorted names are searched for and **num_names** are returned in a character array allocated by the calling program. Along with each name is a spectrum identification number which can be used to retrieve the corresponding spectrum in the main library or in the user library; a name position number in the range **0..last_name_pos** is provided to support scroll bar positioning.

NISTMS_INC_NAME_INFO		
Type	**Field Name**	**Description**
Input information only		
int	**alpha_only**	if non-zero only A-Z allowed and prefixes are ignored
int	**num_names_desired**	maximum number of names returned
int	**one_name_len**	length assigned to each name, longer names are truncated
Output information only		
char*	**names**	allocated by user, filled with found NULL-terminated names; allocated length = **num_names_desired** * **one_name_len**
int	**num_names**	filled with number of actually found names
unsigned long*	**id_nums**	allocated by user, filled with ID numbers for Main NIST library or user library, length = **num_names_desired** * sizeof(long)
int*	**name_pos**	allocated by user, filled with position numbers for MAIN NIST library or user library, length = **num_names_desired** * sizeof(int)
int	**last_name_pos**	filled with the position number for the last name in the MAIN NIST library or user library
char*	**name_key**	allocated by user, filled with name string created out of the compound name pointed to by **string_in** upon request with *search_type* = NISTMS_INC_GET_NAME_KEY according to **alpha_only** value. Allocated length > 18 bytes.

NULL-terminated library names are returned in **names** in adjacent buffers of length **one_name_len**. That is, name **i** (**i** ≥ 0) begins at location **names[one_name_len * i]**. The number of names actually returned is passed in **num_names**, which is equal or less than **num_names_desired.**

18) NISTMS_PEAK_INFO is a data structure used for "anypeak" searches. After initializing this search, each peak is separately processed by submitting its mass (m/z), maximum abundance (or rank), minimum abundance (or rank) and peak type. After each peak is submitted, numbers of matching spectra are returned. After all peaks have been entered, spectra locations are returned in the **NISTMS_HIT_LIST** structure.

NISTMS_PEAK_INFO		
Type	**Field Name**	**Description**
Input: Individual peak specs for NISTMS_ANYPEAK_ONE_PEAK_SRCH		
int	**mass**	mass or neutral loss for the peak
int	**abmin**	minimum abundance or rank of the peak
int	**abmax**	maximum abundance or rank of the peak
NISTMS_PEAK_TYPE	**type**	type of the peak
Output of NISTMS_ANYPEAK_ONE_PEAK_SRCH		
long	**num_for_peak**	number of spectra found for the peak
long[NISTMS_MAXANYPEAKS]	**net_num_matches**	numbers of hits for different numbers of specified peaks in the hits
Input for NISTMS_ANYPEAK_GET_HITS_SRCH		
int	**num_matches_required**	required minimum number of matching peaks in each of the hits

Four types of peaks may be specified using the following enumerated type:

typedef enum {
NISTMS_ANY_PEAK,
NISTMS_LOSS_PEAK,
NISTMS_MAXMASS_PEAK,
NISTMS_AM2_PEAK, /* for internal use only */
NISTMS_RANK_PEAK,
NISTMS_EXACT_MASS_PEAK /* for use with ExactPeak only */
} **NISTMS_PEAK_TYPE;**

num_for_peak gives the number of spectra having the most recently entered peak and **net_num_matches[i]** holds the number of spectra having i ($i \geq 1$)specified peaks.

num_matches_required is used after all peaks have been entered. It gives the minimum number of peaks required for a hit. Actual locations of spectra are returned in a structure of type **NISTMS_HIT_LIST.**

For *NISTMS_EXACT_MASS_PEAK*, instead of three members of **NISTMS_PEAK_INFO**, namely, **mass, abmin, abmax,** the following four members of **NISTMS_PEAK_INFO::ExactPeak**[11] are used

[11] NISTMS_PEAK_TYPE members *mass*, *abmin*, *abmax* occupy the same memory as ExactPeak.

*NISTMS_PEAK_INFO::ExactPeak **		
Type	**Field Name**	**Description**
Input: Individual peak specs for NISTMS_ANYPEAK_ONE_PEAK_SRCH, type= NISTMS_EXACT_MASS_PEAK		
unsigned	**exact_mw_min**	min. mass (f32 format)
unsigned	**exact_mw_max**	max. mass (f32 format)
short	**abmin**	minimum abundance
short	**abmax**	maximum abundance

The first two members may be calculated either with *search_type* = NISTMS_GET_EXACT_MASS_LIMITS and contents of *string_in* similar to those described for *search_type* = NISTMS_EXACT_MASS_SRCH (with the exception of item **`i`**, which should not be used), or converted to f32 format from exact mass limits expressed in Da as double.

19) NISTMS_AUX_DATA contains non-spectral information associated with a library entry. This includes various compound identification information, including chemical name, formula, synonyms, contributor or comment, and nominal molecular weight. The chemical structure is held in a separate structure in NISTMS_IO. Certain information is only returned if a pointer to a buffer (and length when appropriate) is provided by the calling procedure (non zero). Eliminating the retrieval of unnecessary data can help improve performance.

#define NISTMS_MAXNAMELEN 512
#define NISTMS_MAXFORMLEN 24

NISTMS_AUX_DATA		
Type	**Field Name**	**Description**
NISTMS_RECLOC	**spec_loc**	location of spectrum
NISTMS_RECLOC	**stru_loc**	location of chemical structure
unsigned long	**ident**	library identification number, unique for each spectrum in a library
long	**casno**	Chemical Abstracts Service Registry Number
long	**specno**	NIST Accession number
unsigned int	**mw**	Nominal molecular weight (uses most abundant isotopes)
unsigned int	**exact_mw**[2]	Monoisotopic exact mass (f32) or zero
int	**other_dbs**	bitmap of other chemical lists containing this compound
char[NISTMS_MAXNAMELEN]	**name**	chemical name
char[NISTMS_MAXFORMLEN]	**formula**	chemical formula (Hill sorted)
unsigned int	**spec_flags**[2,3]	see NISTMS_SPECTRUM_FLAGS
unsigned char	**instr_type**[2]	MS/MS instrument type, NISTMS_INSTR_TYPE
signed char	**charge**[2]	MS/MS precursor charge
Spectrum contributor or may be used as comment field in user library		
char*	**contributor**	only filled if allocated (non NULL)
int	**contributor_len**	length of allocated contributor buffer, up to 2048 (NISTMS_MAXCONTRIBLEN)
Alternate chemical names as concatenated ASCII null-terminated strings ends with double null		
char*	**synonyms**	only filled if allocated (non NULL)
int	**synonyms_len**	allocated length of synonyms
int	**num_synonyms**	number of synonyms
Locations of other spectra for this molecule		
int	**num_rep_locs**	input: allocated number of longs in rep_loc. Output: number of replicate spectra found
NISTMS_RECLOC *	**rep_locs**	if non-NULL, replicate spectra file location are put here
Peptide MS/MS spectra source references		
char *	**references**[1]	only filled if allocated (non NULL)
int	**references_len**[1]	length of allocated references buffer
int	**num_references**[1]	number of references returned

[1] Available in version 2.1.1; may be read from a peptide MS/MS library only
[2] Available in version 2.1.5.
[3] Present in NIST 12 nist_msms2 library; notes location of Peptide sequence and Peptide modifications in NISTMS_AUX_DATA.

In **references** are returned **num_references** zero-terminated strings containing information about the source of a spectrum in peptide MS/MS library. Each string contains up to seven tab-delimited fields, namely, Dataset, Contributor, Number of Files, Source, Reference, Title, Authors. Internet links, if present, are in curly braces {}.

The **other_dbs** field contains bitmap that denotes the presence of the compound in other chemical databases. These bit positions and databases are:

Bit position	Bit mask	DB type	DB name
1	0x0001	**Fine**	Fine Chemicals (Commercially Available Fine Chemical Index)
2	0x0002	**TSCA**	Toxic Substances Control Act Inventory
3	0x0004	**RTECS**	Registry of Toxic Effects of Chemical Substances
4	0x0008	**EPA**	EPA Environmental Monitoring Methods Index
5	0x0010	**USP**	U. S. Pharmacopoeia/U.S.A.N.
6	0x0020	**HODOC**	CRC Handbook of Data of Organic Compounds
7	0x0040	**NIH**	NIH-NCI Medicinal Chemistry Listing
8	0x0080	**EINECS**	European Inventory of Environmentally Significant Chemical Substances
9	0x0100	**IR**	NIST/EPA Gas Phase Infrared Database

20) INTERP_MS is a data structure contains output information for Chlorine-Bromine Estimation Search, MW Estimation Search, Substructure Information Search .

INTERP_MS		
Type	**Field Name**	**Description**
Following data structure used only for **Chlorine-Bromine Estimation**		
ClBrStruct	**ClBr**	Output data structure for Chlorine-Bromine Estimation
The following fields used only for **MW Estimation**		
int[5]	**mw_est**	5 possible MW values
int[5]	**mw_est_prob**	probabilities for calculated MW values
int[5]	**mw_si**	similarity numbers
The following fields used only for **Substructure Information Search**		
int	**num_substrus**	number of substructures
int[NUM_SUBS]	**substru_prob**	probabilities presence/absence
int[NUM_SUBS]	**OffsetInNameFile**	- not used -

The following table describes the **ClBrStruct data structure .**

ClBrStruct		
Type	**Field Name**	**Description**
int	**nCl**	predicted number of Cl atoms
int	**nBr**	predicted number of Br atoms
int	**prob**	probability that cl and br above are correct
int	**prob_clbr**	probability that compound has Cl or Br
int	**any_cl**	probability that compound has Cl
int	**any_br**	probability that compound has Br
int	**nWarnings**	number of warnings from Cl-Br estimation
char[8][70]	**Warning**	warnings from Cl-Br estimation
char[40]	**Error**	error message from Cl-Br estimation

21) NISTMS_USER_STRUCT_INFO is used for interactive chemical structures scanning, enabling the user to select a desired structure from a list. **max_structs_desired** structures are searched for and **num_structs_found** structures names are returned in a **struct_names** buffer allocated by the calling program. Along with each name is a structure location which can be used to retrieve the structure from the user library or MOL/SDfile, and an sequence number of the structure in the file.
A positive sequence number of the first structure to scan should be provided in the **NISTMS_IO** structure member *string_in*. If *molfile_handle* is not zero, then previously opened MOL/SDfile will be scanned, otherwise the active user library referenced by *active_libs*[0] will be scanned.

NISTMS_USER_STRUCT_INFO		
Type	**Field Name**	**Description**
Input information only		
int	**max_structs_desired**	number of structures to find
int	**max_one_struct_name_len**	length assigned to each name, longer names are truncated
char*	**name_filter**	substring to be found in structures names or NULL
Output information only		
char*	**struct_names**	allocated by user, filled with found names; allocated length ≥ **max_one_struct_name_len** * **max_structs_desired**
int	**num_structs_found**	filled with number of actually found structures
NISTMS_RECLOC*	**stru_pos**	file offsets (starting from 1) to chemical structures, allocated length ≥ **max_structs_desired** × sizeof(NISTMS_RECLOC) Values may be used later with NISTMS_STRU_SRCH
long*	**stru_seq_nums**	sequence numbers of structures, starting from 1.Allocated by user, length ≥ **max_structs_desired** x sizeof(long)

Structure names are returned in **struct_names** in adjacent buffers of length **max_one_struct_name_len**. That is, name number **i** ($0 \le$ **i** $<$ **num_structs_found**) begins at location **struct_names[max_one_struct_name_len** × **i]**. The number of names actually returned is passed in **num_structs_found**, which is equal or less than **max_structs_desired.** If the results refer to the MOL/SDfile, closing and reopening the file does not make the results invalid unless the file has been altered at the file position before the last position in **stru_pos**.

22) molfile_handle is used to access user supplied MOL/SDfiles containing chemical structures. The file can be opened with *search_type* = NISTMS_OPEN_MOLFILE, *string_in* pointing to the full pathname of the file. The file MUST be closed later with *search_type* = NISTMS_CLOSE_MOLFILE, which sets **molfile_handle** to zero**.**

Warning: Non-zero **molfile_handle** value will force the software to read the structures from the open MOL/SDfile only (that is, ignoring all libraries) in cases of *search_type* = NISTMS_SCAN_USER_STRU_SRCH, NISTMS_STRU_SRCH, and NISTMS_ADD_TO_LIBRARY_SRCH.

MISCELLANEOUS

Spectrum and Structure Pointers

Each spectrum pointer is associated with a single library. The zero-based sequence number of this library (in *lib_paths*) may be obtained from the spectrum pointer (*spec_loc*) with the macro NISTMS_LIB_NUM (*spectrum_pointer*). In 2GB API, the actual offset location in the data file (in main library NIST.DB, replicate library NIST.DBR, or user library USER.DBU) is defined by two NISTMS_RECLOC members: *spectrum_pointer.loc* and *spectrum_pointer.ftype*. In previous to 2GB API versions, where *spectrum_pointer* is a 32-bit integer the actual offset location in the data file may be obtained from the lower 7 half-bytes (*spectrum_pointer* & 0x0FFFFFFF).

Structure offset and the sequence number of the library can be obtained from the structure pointer *stru_loc* the same way; with two exceptions: (1) for user libraries and MOL/SDfiles the offset is counted in text mode and 1 is added to the offset so that it is always positive; (2) for MOL/SDfiles all 8 half-bytes of the structure pointer are used to store the file offset in case of previous to 2GB API versions.

User Library Creation

It is the responsibility of the application program to create an empty user library before adding spectra to it. For more detail see section XVI, Creating New User Libraries.

Naming Conventions

All defined structures and values begin with **NISTMS_** and most of *search_type* names end in **_SRCH**.

C/C++ Structure Alignment

In any code that uses structures defined in NISTMS.H, your compiler should be set to align structures defined in NISTMS.H along 1 or 2 byte boundaries. Under Microsoft Visual Studio 6.0 and later, this is achieved by means of lines #pragma pack(push, 1) and #pragma pack(pop) located in NISTMS.H and enclosing its contents.

Special Characters in Compound Names

Most of non-ASCII characters in chemical names and synonyms are represented in Extended ASCII encoding; for instance, Greek letter alpha (α) has extended ASCII code 224 in code page 437. If your font provides correct characters for all extended ASCII codes in the table below you may use these codes directly in displays. Otherwise, or for printing purposes, we recommend converting them into corresponding Unicode characters or substituting them with the corresponding dot-delimited ASCII names, e.g. ".alpha.". The following table provides non-ASCII characters, their Unicode codes, .text. representation, and extended ASCII codes of all characters whose Extended ASCII codes may occur in chemical names and synonyms:

Letter	α	β	γ	δ	ε	π	σ	μ	ω	±	η
Unicode	03B1	03B2	03B3	03B4	03B5	03C0	03C3	03BC	03C9	00B1	03B7
.text.	.alpha.	.beta.	.gamma.	.delta.	.epsilon.	.pi.	.sigma.	.mu.	.omega.	.+/-.	.eta.
ASCII dec	224	225	231	235	238	227	229	230	234	241	252
ASCII hex	E0	E1	E7	EB	EE	E3	E5	E6	EA	F1	FC

A few special characters not included in the table are represented in dot-delimited form (for instance, .DELTA. and .psi., for uppercase delta (Δ) and lowercase psi (ψ), respectively).

Tagged Synonym Display

4 character tags $:nn at the beginning of a chemical name synonym are displayed by the MS Search according to the following table. The tagged strings are displayed in spectrum text information window above the mass spectrum peaks. It is the software developer's responsibility to display these tags as described below.

Tag **Displayed string**

```
$:00    "Spectrum type:"
$:01    "Compound type:"
$:02    "Ion name:"
$:03    "Precursor type:"
$:04    "Precursor m/z:"
$:05    "Collision energy:"
$:06    "Instrument type:"
$:07    "Instrument:"
$:08    "Special fragmentation:"
$:09    "Sample inlet:"
$:10    "Ionization:"
$:11    "Ion mode:"
$:12    "Collision gas:"
$:13    "Pressure:"
$:14    "Mass range:"
$:15    "Maximum intensity:"
$:16    "In-source voltage"   // previously "Cone voltage:"
$:17    "AUX:"
$:18    ""   //link, contains precursor spectrum NIST r.n.; is not displayed
$:19    "Ion Formula:"
$:20    "Ion MW:"
$:21    "Charge:"
$:22    "Salt:"
$:23    "Known impurity:"
// version 2.1.5, NIST 11
$:24    "Related CAS#"
$:25    "Salt/Mix CAS#"
$:26    "Peptide sequence"
$:27    "Peptide mods"
```

For example, the synonym displayed by NIST MS Search in a separate line as "Precursor m/z:567.89" is "$:04567.89" where $:04 is the tag. It is followed by a string to be displayed, 567.89

Tagged synonyms are present in NIST 11 libraries MAINLIB, NIST_MSMS, and NIST_MSMS2.

File nistms_syntag.h contains a table of synonym tags sorted in alphabetic order (case-insensitive.)

Tandem Mass Spectra Representation Conventions *

Currently, there are the following formats for representing tandem mass spectra in a library. They may be found in

(a) NIST 12 library nist_msms (small molecules)
(b) NIST 12 library nist_msms2 (peptides)
(c) NIST peptide libraries available from http://peptide.nist.gov
(d) In-source spectra, which do not have a well-defined precursor m/z

See also comments to NISTMS_MASS_SPECTRUM in the DATA STRUCTURES section.

Precursor m/z *

When adding a spectrum to a library, all formats require non-zero NISTMS_IO::*userms.precursor_ion_100mz*.

(a-c) require precursor m/z greater than 1. Precursor m/z may be saved as an integer value, `(unsigned)floor(100*PrecursorMz+0.5)`, or as a value in f32 format, `nistms_dbl_to_f32(PrecursorMz)`, where `PrecursorMz` has type double and contains actual precursor m/z value.

(d) requires precursor m/z = 0.75, that is, *userms.precursor_ion_100mz*=75

Precursor m/z is the only mandatory member of tandem mass spectra as compared to EI spectra.

(a) NIST 12 MS/MS Small Molecules Spectra *

Use nist_msms spectra to find out typical contents of $:nn synonyms. Note that charge is not allowed in a chemical formula. Below are several examples. M, Cat (cation), or An (anion) stand for the Formula.

Compound type	Precursor type	Comment
Cat2+	[Cat]2+	Precursor ion formula is in Formula; its charge=2, Precursor charge =+2.
Cat2+2Cl-	[Cat+Cl]+	Compound is a chloride; Formula describes cation; its charge=+2. Precursor ion formula is Formula with added atom Cl; Precursor charge=+1.
M	[2M+H]+	Precursor ion is a protonated dimer; its charge=+1
M	[M-H-NH3]-	Precursor ion is a result of NH3 and proton loss; charge=-1
M	[194Pt(NH3)2(37Cl)OH+H]+, [196Pt(NH3)2(35Cl)OH+H]+	Precursor ion is an isotope with m/z=283.00 of [Pt(NH3)2ClOH+H]+; charge=+1
M	[M-73]+	Precursor ion m/z is created by a loss of 73; its chemical formula is unknown; charge=+1

- + or – inside the Compound type are always charges; no losses, additions, or brackets are allowed.
- + or – inside the square brackets of Precursor type may mean only additions or losses, respectively;

the only charge is the total charge outside of the brackets. Parentheses are allowed. Currently, isotopic formulas cannot be checked by Lib2NIST.

- Cat or An are assumed uncharged; that is, their charges are always explicitly shown.
- If precursor charge is known, add synonym $:21 or set NISTMS_AUX_DATA::*charge*
- If Instrument type is known, add synonym $:06 or set NISTMS_AUX_DATA::*instr_type*

If instrument type string is the same (ignoring letter case) as in the following table, the dll overwrites *instr_type:*

Instrument type string (synonym $:06)	***instr_type*** [1]	**value**
"IT/ion trap" or "QQIT" or "QqLIT"	NISTMS_INSTR_TYPE_IONTRAP	1
"Q-TOF" or "HCD"	NISTMS_INSTR_TYPE_QTOF	2
"QqQ/triple quadrupole"	NISTMS_INSTR_TYPE_QQQ	3

[1] see NISTMS_INSTR_TYPE in NISTMS.H

If instrument type string exists and is not in the table, *instr_type* is set to INSTR_TYPE_UNK = 31. If instrument type string does not exist, *instr_type* is used without checking if it is valid; note that INSTR_TYPE_NONE = 0 (no instrument type is available). Saved in the spectrum record *instr_type* is used in Instrument type constraint, see NISTMS_BIT_INSTR_TYPE in NISTMS.H.

(b) NIST 12 MS/MS Peptide Spectra *

New $:nn tagged synonyms, "Peptide sequence" and "Peptide mods" are used. Section "(a) NIST 12 MS/MS Small Molecules Spectra" applies these spectra. Use nist_msms2 library spectra to find out typical contents of $:nn synonyms. Product ion peaks are annotated similarly to NIST Peptide libraries.

(c) Spectra in NIST peptide libraries *

See http://chemdata.nist.gov/mass-spc/ftp/mass-spc/PepLib.pdf , Section Appendices (pp.12-17.)
Unfortunately, this document has not been updated since 1996.
In NIST peptide library spectra, all auxiliary information is in the Name (which is peptide sequence/Charge) and Comment. Charge may be negative. Product peaks are annotated; these annotations are used by Peptide MS/MS search.

(d) In-source spectra *

Should not have "Precursor type" and "Precursor m/z" synonyms; have *precursor_ion_100mz*=75.
Section "(a) NIST 12 MS/MS Small Molecules Spectra" applies to these spectra.

Limitations

1) No more than 16 libraries may be accessed (*num_libs* in NISTMS_IO can be no greater than 16) except in 2GB API version, where up to 127 libraries may be accessed.

2) The unique spectrum ID values in the user library created by the NIST MS dll cannot exceed 65535; the ID of the newly added spectrum is calculated by adding 1 to the greatest spectrum ID present in the library. This imposes limitation on the maximum number of spectra in the user-created library.

3) There may be no more than 800 (NISTMS_MAXPEAKS) peaks in an EI spectrum (however, the number of text peaks in a MS/MS spectrum may exceed 800 and reach 7,000-8,000, see NISTMS_DFLT_MAX_PEAK_TXTDATA_NUM in NISTMS.H). There is an upper limit for the size of text peaks character array, NISTMS_DFLT_MAX_PEAK_TXTDATA_LEN in NISTMS.H.

4) The largest number of spectrum pointers in NISTMS_HIT_LIST is 6000 (NISTMS_MAX_FPOS).

5) When performing a library search, the smallest number of *max_spectra_desired* in NISTMS_HIT_LIST is 100 (MAX_LIB_SRCH_HITS).

6) All masses in EI spectra are represented as integers. Peaks having fractional masses should be rounded to integers and two peaks having the same mass should be represented as a single peak (usually by ignoring all but the most intense peak). The MS/MS spectrum "exact" (text) peaks are not subject to this restriction. However, when filling out a MS/MS NISTMS_MASS_SPECTRUM, integer peaks usually must be also present. ReadMSP.c has code illustrating how to convert peaks with non-integer m/z and abundances into peaks with integer m/z and abundances.

7) Due to the presence of static variables, the NIST MS DLL cannot be reliably used by more than one thread in an application at a time.

8) The .NET interface to NIST MS Dll, NISTMSCL, requires code page 1252 be available (but not necessarily active) in Windows.

Calling Conventions

All NIST DLL functions described in this document follow C calling conventions (_cdecl) if their names contain lowercase letters (nistms_search(), CreateUserLibrary(), GetNumberOfEntries(), and GetLibraryType()).

Delphi and Visual Basic programmers may call all DLL functions using their UPPERCASE names: NISTMS_SEARCH(), CREATEUSERLIBRARY(has reversed order of arguments), GETNUMBEROFENTRIES(), and GETLIBRARYTYPE(); these functions have __stdcall (pascal) calling conventions (see file nistms.h file for details).

If NISTMS_SEARCH() is called, then the *callback* function pointer should refer to a __stdcall or pascal function. (The *callback* is used only for *search_type* = NISTMS_INIT_SRCH)

"SEARCH TYPE" SUMMARY TABLE

Input and output information associated with *search_type* are described below:

"SEARCH TYPE" SUMMARY		
Search Type	**Input**	**Output**
NISTMS_SET_VERSION	*String_in* = "2.1.1" or "2.0"	- none -
NISTMS_INIT_SRCH [N]	num_libs, *lib_paths*, *lib_types*, callback[O]	-none-
NISTMS_CLOSE_SRCH [N]	-none-	-none-
NISTMS_GET_SPECTRUM_SRCH [N]	input_spec_loc **libms** (*exact_mz*, exact_mz_len, *buf_exact_mz*, buf_exact_mz_len)[O], **aux_data**[O] (contrib_len[O],synonyms_len[O], num_reps_loc[O], references_len[O])	**libms** (num_peaks, mass, abund, precursor_ion_100mz[O], num_exact_mz[O], *exact_mz*[O], *buf_exact_mz*[O]), **stdata**[O], **aux_data**[O] (*contributor*[O], *synonyms*[O], *rep_loc*[O], *references*[O], spec_loc, stru_loc, ident, casno, specno, mw, other_dbs, name, formula, num_synonyms[O], num_reps_loc[O], num_references[O])
NISTMS_GET_STRUCTURE_SRCH	left in for backward compatibility only	
~~NISTMS_GET_SYNONYMS_SRCH~~	- eliminated -	
Get one spectrum		
NISTMS_NAME_SRCH [S]	*string_in*	output_spec_loc
NISTMS_ID_SRCH [S]	*string_in* (single ID like "5")	output_spec_loc
ANYPEAKS search		
NISTMS_ANYPEAK_INIT_SRCH	-none-	-none-
NISTMS_ANYPEAK_ONE_PEAK_SRCH	**peak_info** (mass, abmin, abmax, type)	**peak_info**(num_for_peak,net_num_matches)

"SEARCH TYPE" SUMMARY		
Search Type	**Input**	**Output**
NISTMS_ANYPEAK_GET_HITS_SRCH	**peak_info**(num_matches_required) **hit_list** (max_spec_locs)	**hit_list** (*spec_locs*, num_hits_found)
Get set of hits		
NISTMS_INC_FIRST_NAME_SRCH [S]	*string_in*, **name_info**(num_names_desired, one_name_len, alpha_only)	**name_info** (*names*, *id_nums*, *name_pos*, last_name_pos)
NISTMS_INC_NEXT_NAME_SRCH [S]	*string_in*, **name_info**(num_names_desired, one_name_len, alpha_only)	**name_info** (*names*, *id_nums*, *name_pos*, last_name_pos)
NISTMS_INC_PREV_NAME_SRCH	left in for backward compatibility only	
NISTMS_CASNO_SRCH, NISTMS_CASNO_SRCH2	*string_in*, **hit_list**(max_spec_locs)	**hit_list** (*spec_locs*, num_hits_found)
NISTMS_NISTNO_SRCH	*string_in*, **hit_list**(max_spec_locs)	**hit_list** (*spec_locs*, num_hits_found)
NISTMS_EXACT_MASS_SRCH	*string_in*, **constraints**° **hit_list**(max_spec_locs)	**hit_list** (*spec_locs*, num_hits_found)
NISTMS_MW_SRCH	*string_in*, **hit_list**(max_spec_locs)	**hit_list** (*spec_locs*, num_hits_found)
~~NISTMS_REP_SRCH~~	- eliminated -	
NISTMS_FORMULA_SRCH	*string_in*, **hit_list**(max_spec_locs)	**hit_list** (*spec_locs*, num_hits_found)
NISTMS_SEQ_ID_SRCH [S]	*string_in* (1st call only, usually "1"), **hit_list**(max_spec_locs)	**hit_list** (*spec_locs*, num_hits_found)
NISTMS_ID_SRCH [S]	*string_in* (ID range like "1-5"), **hit_list**(max_spec_locs)	**hit_list** (*spec_locs*, num_hits_found)
User spectrum library search		
NISTMS_SCREEN_SRCH	**userms, cntls**	**hit_list** (*spec_locs*)

"SEARCH TYPE" SUMMARY		
Search Type	**Input**	**Output**
NISTMS_COMPARE_SPECTRA_SRCH [N]	**userms**, **cntls**, **constraints**[O] **hit_list** (num_hits_found, *spec_locs*, lib_names_len, max_one_lib_name_len, max_hits_desired, max_spec_locs)	**hit_list** (*spec_loc*, *sim_num*, *rev_sim_num*, *hit_prob*, *lib_names*[O], *stru_pos*[O], *casnos*[O], *pep_Mf*[OM] *pep_hdr_Mf*[OM])
NISTMS_NO_PRE_SRCH	**userms**, **cntls**, **constraints**[O] **hit_list** (num_hits_found, *spec_locs*, lib_names_len, max_one_lib_name_len, max_hits_desired, max_spec_locs)	**hit_list** (*spec_loc*, *sim_num*, *rev_sim_num*, *hit_prob*, *lib_names*[O], *stru_pos*[O], *casnos*[O], *pep_Mf*[OM] *pep_hdr_Mf*[OM])
NISTMS_CL_BR_EST [N]	**userms**	**interp_ms**(ClBr)
NISTMS_MW_EST [N]	**userms**, **hit_list** (num_hits_found, *sim_num*, *spec_locs*)	**interp_ms**(*mw_est[]*, *mw_est_prob[]*, *mw_si[]*)
~~NISTMS_MW_ESTIMATION_2~~	- eliminated -	
NISTMS_SUBSTR_SRCH [N]	**hit_list** (num_hits_found, *sim_num*, *casnos*)	**interp_ms**(num_substrus, *substru_prob[]*)
Convert list of spectrum locations to compound identification information for display		
NISTMS_BUILD_HITLIST_SRCH [N]	**hit_list**(num_hits_found, *spec_locs*, lib_names_len , max_one_lib_name_len, max_spec_locs, max_hits_desired) **constraints**[O]	**hit_list** (*spec_locs*, *lib_names*[O], *stru_pos*[O], *casnos*[O])
User library maintenance		
NISTMS_ADD_TO_LIBRARY_SRCH [S]	**userms**, **aux_data**, molfile_handle[O]	**aux_data** (ident)
NISTMS_DELETE_FROM_LIBRARY_SRCH [S]	*string_in*	- none -

"SEARCH TYPE" SUMMARY		
Search Type	**Input**	**Output**
Handling user defined chemical structures		
NISTMS_STRU_SRCH [SN]	input_spec_loc[O], molfile_handle[O], **aux_data** (stru_loc[O], spec_loc[O], ident[O] and *active_lib*[0][O], casno[O] or specno[O])	**stdata**
NISTMS_OPEN_MOLFILE [N]	*string_in*	molfile_handle < 0
NISTMS_CLOSE_MOLFILE [N]	molfile_handle	molfile_handle = 0
NISTMS_SCAN_USER_STRU_SRCH [SN]	*string_in*, molfile_handle[O], **user_struct_info** (max_structs_desired, *name_filter*[O], max_one_struct_len)	**user_struct_info** (*struct_names*, *stru_pos*, *stru_seq_nums*, num_structs_found)
Miscellaneous		
NISTMS_INDEX_USER_STRU	- none -	- none -
NISTMS_INDEX_LIBRARY_NAMES [S]	- none -	- none -
NISTMS_INDEX_LIBRARY_EXACT_MASS [S]	- none -	- none -
NISTMS_GET_EXACT_MASS_LIMITS [N]	*string_in*, **peak_info**	**peak_info** (ExactPeak.exact_mw_min, ExactPeak.exact_mw_max)
NISTMS_INC_GET_NAME_KEY [N]	*string_in*, **name_info**(alpha_only)	**name_info** (*name_key*)
NISTMS_DECODE_MODS [N] (decode compressed peptide modifications)	**aux_data**(*contributor*, contributor_len)	**aux_data**(*contributor*)
NISTMS_MAKE_MOLFILE [N]	*string_in*, **stdata**, aux_data=NULL	Molfile with name from string_in, structure from stdata
NISTMS_MAKE_MOLFILE [N]	*string_in*, **aux_data**(stru_loc), stdata=NULL	Molfile with name from string_in, copied from user library according to stru_loc

"SEARCH TYPE" SUMMARY		
Search Type	**Input**	**Output**
NISTMS_ALT2AROM [N]	*string_in*, **stdata**	**stdata**
NISTMS_MARK_LIBS	*string_in*, **aux_data**	**aux_data**.name
NISTMS_MARK_ALL_LIBS	*string_in*, **aux_data**	**aux_data**.name
NISTMS_EXACT_MZ_TO_INT_PEAKS [N]	**userms**(*buf_exact_mz*, *exact_mz*, num_exact_mz)	**userms**(mass, abund, num_peaks)

NOTES:

1) *active_libs* specifies one or more search libraries except for search types marked with superscripts S and/or N:.

- superscript S ([S]) indicates that *active_libs* must specify at least one search library; only the first specified library will be used.
- superscript N ([N]) indicates that *active_libs* are ignored.
- superscript SN ([SN]) indicates that depending on the context none or only the first specified in *active_libs* library is used.
- superscript M ([M]) indicates pointers to memory allocated for MS/MS search.

2) **bold** denotes pointers to "C"-structures defined in NISTMS.H

3) *italics* denotes arrays to be allocated by calling program

4) superscript O ([O]) indicates that use is optional.

5) variables in parentheses are members of the preceding structure pointer

6) all structures must be allocated by the calling program.

UPDATING FROM PRIOR VERSIONS

Breaking Changes in version 2.1.5*

In general, rebuilding your program with version 2.1.5 of the dll should not change the functionality of your program with a single exception: NISTMS_MASS_SPECTRUM member *precursor_ion_100mz* is returned in f32 format. To avoid this, initiate the dll with the "2.1.4" string instead of "2.1.5". However, this may reduce precursor m/z accuracy retrieved from a library spectrum from 0.0001 to 0.01 m/z units.

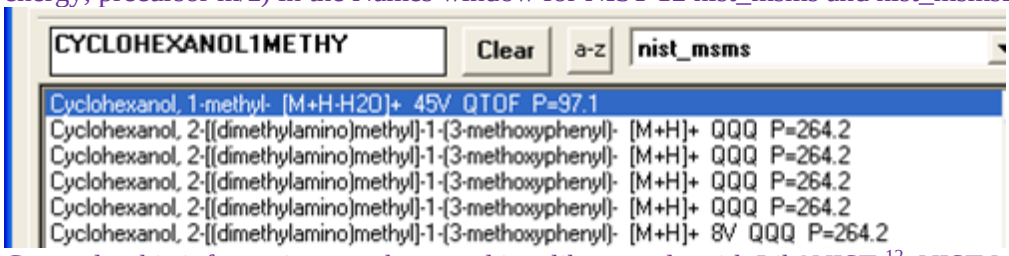
Alternatively, convert NISTMS_MASS_SPECTRUM::*precursor_ion_100mz* into a double using the definition of NISTMS_PRECUR_MZ_TO_DBL(X) from NISTMS.h:

```
double dPrecursorMz = NISTMS_PRECUR_MZ_TO_DBL (mass_spec->precursor_ion_100mz);
```

This macro correctly converts both f32 and integer representation used in previous versions.

Changes to NIST MS/MS 12 Incremental Name Search Information *

NIST MS Search displays MS/MS-related information (precursor type, instrument type, collision energy, precursor m/z) in the Names window for NIST 12 nist_msms and nist_msms2 libraries:



Currently, this information may be stored in a library only with Lib2NIST [12]. NIST MS Dll is unable to do this. Therefore, do not index names in NIST 12 MS/MS Libraries, nist_msms and nist_msms2, with the NIST MS Dll otherwise MS/MS information displayed in the Names window will be lost.

Changes in case of negative *min_mass* and *max_mass* (NISTMS_SRCH_CONTROLS)

In previous versions, MS/MS or Peptide search with values of *min_mass* below -1 was done using absolute values of this parameter or even ignoring it. Starting from v.2.1.5.8, in MS/MS and Peptide searches, its meaning is the same as it is in an EI search.

If m1 is the smallest product peak m/z in the search spectrum, m2 is the smallest product peak m/z in the library spectrum, and min_mass < -1, then peak comparison begins at the m/z = median of 3 values: m1, m2, and –min_mass.

max_mass = -1 means that the highest mass in MS/MS search is approximately 4000.

[12] Version v.1.0.4.16 build 2012-05-15 or later; command line option /MsmsIncNames is required.

New Features in version 2.1.5*

UP TO 127 LIBRARIES. Starting from September 2012 (version 2.1.5), the 2GB API supports initiation and searching of up to 127 libraries. Previous limit, 16 libraries, is still in ¼ GB API.

EXACT MASS SUPPORT. Starting from September 2012 (version 2.1.5), the DLL supports searching libraries for exact molecular mass and exact molecular mass constraint. Some of NIST 11 libraries (Mainlib, Replib, nist_msms) allow for Any Peaks Exact Mass search.

NEW CONSTRAINT TYPES. Starting from September 2012 (version 2.1.5), the DLL supports constraints by exact mass and (for MS/MS searches) instrument type.

NEW MS/MS SEARCH. Starting from September 2012 (version 2.1.5), the DLL supports searching MS/MS libraries with a non-peptide (small molecules) MS/MS search designed for searching small molecules mass spectra (SEARCH_MODE_FLAG_GENERIC_MSMS).

NEW PEAK MATCHING algorithm for MS/MS spectra compare. Starting from September 2012 release (version 2.1.5), the DLL provides an alternative peak matching method, which provides more stable results (SEARCH_MODE_FLAG_ALT_PEAK_MATCHING).

IN-SOURCE TANDEM SPECTRA (December 2012, version 2.1.5) Spectra with accurate peak m/z and intensities, which do not have a well-defined precursor m/z (to search for them, use no presearch MS/MS search)

New Features in version 2.1.3

2GB API. Starting from the June 2010 release (version 2.1.3), an updated NIST MS DLL API was introduced, referred to as 2GB API, which allows to search libraries containing data files up to 2GB each.

LARGE LIBRARY SUPPORT. Starting from June 2010 release (version 2.1.3) the DLL supports searching libraries that have up to 1,048,560 spectra per library (previous versions could search libraries with up to 786,420 spectra per library).

New Features in version 2.1.2

FAST MS/MS AND EI LIBRARY SEARCH. Starting from the August 2008 release (version 2.1.2), a two-step search MS/MS library was added (requires peak_pm0.dbu and peak_pm0.inu files in the library), as well as a fast two-step spectrum search in EI and MS/MS library. The two-step MS/MS search feature may be used only upon version 2.1.1 initiation.

New Features in version 2.1.1

SUPPORT FOR COMMENT/CONTRIBUTOR FIELD SEARCHES. Starting from the June 2006 release (version 2.1.1), the DLL accepts conditions for the Constrained spectrum search and the Sequential search that allow selection of only those spectra whose Comment/Contributor field satisfy specified conditions. The maximum size of this field, NISTMS_MAXCONTRIBLEN, was increased to 2048 bytes. For detail see “Syntax of Tags in Comment and Text Info constraint” section in the Appendix. These features may be used only upon version 2.1.1 initiation.

MS/MS LIBRARY SEARCH. Starting from the June 2006 release (version 2.1.1), a spectrum search in MS/MS library or NIST peptide MS/MS library was added together with peptide search-specific constraints. These search and constraints may be used only for searching a MS/MS spectrum in MS/MS libraries. The peptide-specific constraints may be used only in Sequential search and No presearch peptide search. These features may be used only upon version 2.1.1 initiation.

New Features in version 2.0

Several new features have been added to the user-created MS libraries and to the NIST DLL released in January 1999.

STRUCTURES IN USER LIBRARIES. Starting from the 1999 release, the DLL can add and retrieve structures from the user-created MS libraries as well as scan structures for user selection in the user libraries and user-supplied files. A one-time procedure needed for indexing the structural part of older user libraries is provided. Since the September 2000 release, stereochemical information is extracted from Molfiles or user libraries and stored in previously unused elements of bond types.
INCREMENTAL NAME SEARCH. This search feature has been added to user-created libraries. A one-time procedure for enabling the older library to be searched in this way has also been provided.

ADDITION/DELETION OF THE USER LIBRARIES SPECTRA. This is now significantly faster at the expense of small format changes. Older user libraries are fully compatible with this software; however. For more detail see "differences between old and new user libraries" below.

ID NUMBER SEARCHING. It is now possible to retrieve in one call all spectra within a range of ID numbers.

A new feature has been added to the August 1999 version of the DLL for handling user-created MS libraries:

SUPPORT FOR CHEMICAL NAME SYNONYMS IN USER LIBRARIES. Starting from the August 1999 release, DLL can add and retrieve chemical name synonyms from the user-created MS libraries. Presence of the synonyms in the user library may render it incompatible with versions of the NIST MS Search prior to v1.7.

NO PRESEARCH SEARCH. Starting from the September 1999 release, a new type of user spectrum search was added. Unlike regular search, it compares the unknown spectrum to all spectra in the active libraries. The hit list produced by this search may contain up to 400 spectra.

Summary of Implementation Changes in versions 2.1.5*

1. Added new floating point format, f32, for precursor m/z and exact mass values[13]. f32 variables have type unsigned int. All f32 members except precursor_ion_100mz are first introduced in version 2.1.5 of the dll.

[13] See Appendix 5: "f32 – a 32-bit Mass and m/z Floating Point Representation (version 2.1.5)"

Data Structure members in f32 format *	
Data structure	**Members in f32 format**
NISTMS_PEAK_INFO::ExactPeak	exact_mw_min
	exact_mw_max
NISTMS_CONSTRAINTS	exact_mw_min
	exact_mw_max
NISTMS_AUX_DATA	exact_mw
NISTMS_SRCH_CONTROLS	precursor_ion_100mz [1)]
NISTMS_MASS_SPECTRUM	precursor_ion_100mz [1,2)]

[1)] an integer value = 100*(precursor m/z) is also allowed on input.
[2)] To avoid output in f32 format, initiate the dll with "2.1.4" instead "2.1.5"

2. Added new members to the following data structures:
   a) NISTMS_AUX_DATA (*exact_mw*, *instr_type*, *charge*);
   b) NISTMS_CONSTRAINTS (*num_top_isotopes*, *exact_mw_min*, *exact_mw_max*, *bits_instr_type*);
   c) NISTMS_PEAK_INFO (*ExactPeak*, located in place of *mass*, *abmin*, *abmax*);
   d) NISTMS_SRCH_CONTROLS (*pep_bRefPeakFraction*).

3. Changed member type from long (32-bit signed integer) to unsigned (32-bit unsigned integer) to allow f32 floating point format in the following data structures:
   a) NISTMS_MASS_SPECTRUM (*precursor_ion_100mz*);
   b) NISTMS_SRCH_CONTROLS (*precursor_ion_100mz*).

4. Changed member type from int to float to allow a greater range of tolerances:
   NISTMS_SRCH_CONTROLS (*precursor_ion_tolerance*, *product_ions_tolerance*)

5. Added enumeration of Instrument Types, NISTMS_INSTR_TYPE, returned in NISTMS_AUX_DATA (*instr_type*).

6. Added enumeration of Instrument Type indicator bits, NISTMS_BIT_INSTR_TYPE, used in NISTMS_CONSTRAINTS (*bits_instr_type*).

7. Added new *search_type* types: NISTMS_MARK_ALL_LIBS, NISTMS_INDEX_LIBRARY_EXACT_MASS, NISTMS_EXACT_MASS_SRCH, NISTMS_GET_EXACT_MASS_LIMITS, NISTMS_EXACT_MZ_TO_INT_PEAKS, NISTMS_CASNO_SRCH2, NISTMS_NISTNO_SRCH.

8. Added new SEARCH_MODE_FLAGS used to modify NISTMS_SRCH_CONTROLS (*search_mode*):
   SEARCH_MODE_FLAG_ALT_PEAK_MATCHING,
   SEARCH_MODE_FLAG_GENERIC_MSMS,
   SEARCH_MODE_FLAG_REJECT_OTHER,
   SEARCH_MODE_FLAG_PRECUR_MZ_TOL_PPM,
   SEARCH_MODE_FLAG_PROD_PEAK_TOL_PPM.

9. Added new NISTMS_PEAK_TYPE, NISTMS_EXACT_MASS_PEAK, for Any Peaks Exact Mass search.

10. Added library test types "3", "4", "5", and "6" to *search_type*= NISTMS_MARK_LIBS; added new *search_type*= NISTMS_MARK_ALL_LIBS, which tests search capabilities of all initiated libraries.

Summary of Implementation Changes in versions 2.1.3.

1. Introduced 2GB API, in which all formerly 32-bit integer library spectrum or structure location variables (*input_spec_loc, output_spec_loc, spec_locs, stru_pos, spec_loc, stru_loc, rep_locs*) have type NISTMS_RECLOC (6 bytes). In the previous API, the type of NISTMS_RECLOC is long.
2. The DLL is able to search most recent NIST peptide MS/MS libraries.
3. Added 2 structure-related search_types: NISTMS_MAKE_MOLFILE and NISTMS_ALT2AROM.
4. Added search_type NISTMS_MARK_LIBS to check whether a ms/ms search is compatible with each of active libraries.
5. ASCII string functions from C library (e.g. strrchr(…)) acting on path and file names have been replaced with their multi-byte versions (e.g. _tcsrchr (…)). As the result, NIST MS Dll accepts multi-byte pathnames created according to the current system Windows ANSI code page out of Unicode pathnames (CP_ACP in WideCharToMultiByte(…)).

Summary of Implementation Changes in versions 2.1.2.

1. Added a two-step MS/MS search.
2. Added a fast two-step search mode by introducing SEARCH_MODE_FLAG_FAST_PRESEARCH.
3. Added functionality provided by SEARCH_MODE_FLAG_IGNORE_PRECURSOR.
4. A minimal requirement for a MS library to be treated as a MS/MS library is that each spectrum in the library must have a precursor m/z value; these values are indexed in the file precmz.inu, which must be present in any MS/MS library.
5. Fixed a typo is structure member names: NISTMS_SRCH_CONTROLS::*product_ions_tolearnce* has been renamed to *product_ions_tolerance*.
6. MS/MS search spectra comparison algorithm has been improved, which resulted in changed MS/MS search scores.

Summary of Implementation Changes in version 2.1.1

1. Added version 2.1.1 features initiation: *search_type* = NISTMS_SET_VERSION while *io->string_in* points to the version string, "2.1.1". To disable this mode use version string "2.0".
2. Comment/Contributor length, NISTMS_MAXCONTRIBLEN, has been increased from 1024 to 2048.
3. Added new members to the following data structures:
   a) NISTMS_MASS_SPECTRUM (*precursor_ion_100mz, num_exact_mz, exact_mz, exact_mz_len, buf_exact_mz, buf_exact_mz_len*);
   b) NISTMS_SRCH_CONTROLS (*precursor_ion_tolerance, product_ions_tolearnce, precursor_ion_100mz, pep_bTF_qry, pep_bE_Omssa, pep_bTF_lib, pep_bOmssa,*

pep_bNumReplicates, pep_bQ_TOF, pep_cThreshold, pep_nCysteineModification);
c) NISTMS_CONSTRAINTS (*comment_tags, pep_name_frag, pep_min_charge, pep_max_charge, pep_min_protons, pep_max_protons, pep_min_residues, pep_max_residues*);
d) NISTMS_HIT_LIST (*pep_Mf, pep_hdr_Mf*)
e) NISTMS_AUX_DATA (*references, references_len, num_references*)

4. Added "no presearch" and "default presearch" MS/MS library searches (*search_type* = NISTMS_NO_PRE_SRCH, *cntls.search_mode* = 'E')
5. Added constraints for Contributor/Comment.
6. Added peptide-specific constraints.
7. 16-bit (Windows 3.1x) DLL version is not supported.

Summary of Implementation Changes in version 2.0

Refer to "SEARCH TYPE" SUMMARY section for a quick reference.

1. The *stru_pos* member of NISTMS_HIT_LIST and *stru_loc* member of NISTMS_AUX_DATA may reference not only the Main NIST library structures, but also user library structures.
2. In a NISTMS_ID_SRCH search, a range may be specified (e.g., 1-100).;
3. The DLL recognizes a special type of very large read-only user libraries (e.g., Wiley Registry);
4. NISTMS_MAX_FPOS (a system limit for number of hits) has been increased from 2000 to 6000
5. New search types added: NISTMS_OPEN_MOLFILE, NISTMS_CLOSE_MOLFILE, NISTMS_SCAN_USER_STRU_SRCH, NISTMS_STRU_SRCH (replaces NISTMS_GET_STRUCTURE_SRCH), NISTMS_INDEX_USER_STRU, NISTMS_INDEX_LIBRARY_NAMES, NISTMS_INC_GET_NAME_KEY;
6. Search type removed: NISTMS_MW_ESTIMATION_2
7. Obsolete search types kept only for backward compatibility: NISTMS_INC_PREV_NAME_SRCH, NISTMS_GET_STRUCTURE_SRCH
8. User libraries can now be searched by NISTMS_INC_FIRST_NAME_SRCH and NISTMS_INC_NEXT_NAME_SRCH (however, backward compatibility has been preserved);
9. Changed and/or added new meanings to 0 and -1 values of *min_mass* and *max_mass* members of NISTMS_SRCH_CONTROLS structure.
10. Search type NISTMS_BUILD_HITLIST_SRCH optionally may fill out *casnos* member of NISTMS_HIT_LIST data structure
11. The MS minimum peak abundance in searches for Spectra With Specified Peaks has been lowered from 5% to 1%; for both NIST and user-created libraries.
12. Member *OffsetInNameFile* of INTERP_MS is for internal use only; it does not return any meaningful information
13. New members were added to the following C structures:
    a) NISTMS_IO (*molfile_handle*, *user_struct_info*),
    b) NISTMS_INC_NAME_INFO (*name_pos*, *last_name_pos*, *name_key*)
    c) NISTMS_SRCH_CONTROLS (*max_mass*)
14. Changed C structures members types:
    a) NISTMS_INC_NAME_INFO(*id_nums*) unsigned int* to unsigned long*

b) NISTMS_PEAK_INFO(*num_for_peak*, *num_net_matches*[ ]) int to long
c) NISTMS_AUX_DATA(*rep_locs*) unsigned long* to long*

15. In the August 1999 release of the software the following defined in NISTMS.H constants have been changed:

Constant	Previous value	New value	Used in
NISTMS_MAXBONDS	200	254	NISTMS_STDATA
NISTMS_MAXCIRCS	200	254	NISTMS_STDATA
NISTMS_MAXSTRINGS	200	254	NISTMS_STDATA
NISTMS_MAXNAMELEN	300	512	NISTMS_AUX_DATA

In addition, max. allocated length *aux_data->contributor_len* of *aux_data->contributor* (defined as NISTMS_MAXCONTRIBLEN) has been changed from 256 to 512 or 1024 in version 2.0 and to 2048 in version 2.1.1.

Bugs Fixed in version 2.1.5 *

1. References were not read from a peptide spectrum.
2. NISTMS_SRCH_CONTROLS member *bRevImpure* was not included in documentation.
3. In CallDll.c example, when calculating integer m/z and abundances out of "exact" m/z and abundances, in case of peaks with equal integer m/z the greatest abundance was erroneously chosen. To fix the bug, these abundances are summed in this version by using *search_type* = NISTMS_EXACT_MZ_TO_INT_PEAKS first introduced in v.2.1.5.
4. In converting INTERP_MS to NISTMSCL::INTERP_MS_CLI, not more than INTERP_MS_CLI::NUM_MW_ESTIMATES_CLI elements were returned in INTERP_MS_CLI member arrays *substru_prob* and *OffsetInNameFile*. After this bug fixed, up to INTERP_MS_CLI::NUM_SUBS_CLI elements are returned.
5. This document several times referred to an incorrect *search_type*=NISTMS_OPEN_SRCH instead of NISTMS_INIT_SRCH.
6. This document several times referred to an incorrect *search_type*=NISTMS_CAS_SRCH instead of NISTMS_CASNO_SRCH.

Bugs Fixed in version 2.1.3, revision May 23, 2011 *

1. In .NET interface, NISTMSCL, *num_strs* and *radpix* always returned equal to zero when reading a structure into NISTMSCL::STDATA using SEARCH_IO::nistms_search(...).
2. In .NET interface, NISTMSCL, contents of NISTMSCL::STDATA were ignored in calls to SEARCH_IO::nistms_search(...)

Bugs Fixed in version 2.1.3

1. A bug in creation and updating MS/MS two-step presearch index files, which leads to missing hits in two-step searching MS/MS libraries created or maintained with the NIST MS dll. It is recommended to rebuild these libraries with Lib2NIST tool. This bug was introduced in version 2.1.2.
2. Lib number was not saved in 4 most significant bits of io->user_struct_info->stru_pos[i] in case of `NISTMS_SCAN_USER_STRU_SRCH` in a user library.

3. Lowercase letter i in the peak annotation of a non-peptide MS/MS library spectrum caused the peak to be ignored during comparison to the search MS/MS spectrum, resulting in missed hits.
4. In .NET interface, NISTMSCL, an active East Asian language code page caused output of wrong Greek letters in chemical names. This bug was introduced in version 2.1.1 of the .NET interface assembly.

Bugs Fixed in version 2.0

The following bugs found in releases prior to 1999 have been fixed:

1. A number of searches which do not need *active_libs* could return error if *active_libs* was not set;
2. *alpha_only* parameter of Incremental Name Search was improperly inverted;
3. *aux_data* member of NIST_IO could not be NULL for *search_type* = NISTMS_GET_SPECTRUM_SRCH
4. without a properly filled out NISTMS_IO member *lib_types*, *search_type*s other than NISTMS_INIT_SRCH could fail
5. *search_type* = NISTMS_ANYPEAK_GET_HITS_SRCH did not return *error_code* on error and always acted as if *peak_info->num_matches_required* was equal to the number of entered peaks..
6. *abmin* and *abmass* members of NISTMS_CONSTRAINTS were interpreted differently from *abmin* and *abmax* members of NISTMS_PEAK_INFO.
7. Wrong identifier of the length of *substru_prob* member of INTERP_MS structure in the documentation.
8. The search type NISTMS_CL_BR_EST used as an input the spectrum referenced by *libms* member of NISTMS_IO instead of *userms*
9. More than 5 libraries could not be searched at a time.

How to update to version 2.1.5 *

The only breaking change is in the format of NISTMS_MASS_SPECTRUM member *precursor_ion_100mz*, which is returned in f32 format. It is recommended to convert NISTMS_MASS_SPECTRUM::*precursor_ion_100mz* into a double using the definition of NISTMS_PRECUR_MZ_TO_DBL(X) from NISTMS.h:

```
double dPrecursorMz = NISTMS_PRECUR_MZ_TO_DBL (mass_spec->precursor_ion_100mz);
```

This macro correctly converts both f32 and integer representation used in previous versions.

To store expressed as double accurate precursor m/z into *precursor_ion_100mz*, use one of the conversion functions exported from the version 2.1.5 of the dll:

```
mass_spec->precursor_ion_100mz = nistms_dbl_to_f32( dPrecursorMz );
```

The same approach may be used to fill out other variables if f32 format (See Table in the section "Summary of Implementation Changes in versions 2.1.5.")

Note that #define INTERNALBUILD is not needed anymore: in this version, it is in NISTMS.H.

How to update to 2GB API included in version 2.1.3

Switching your program from regular (¼GB) NIST MS Dll API to 2GB API is straightforward:

1. Change the type of NIST MS Dll API file offset variables (see Appendix 4 for 2GB API NISTMS_RECLOC members) used in your application from long to NISTMS_RECLOC.
2. Make sure NISTMS_6BYTE_RECLOC preprocessor definition is present among C/C++ compiler command line options used to build your application.
3. Make sure NISTMS.H header is included in all source files that refer to file offset variables obtained from NIST MS Dll or used in it.
4. Change format of data files where your application saves NIST MS Dll files offsets according to sizeof(NISTMS_RECLOC) = 6 instead of sizeof(long)=4.
5. Link your application with nistdlXX_2gb.lib (XX=32 or 64) instead of nistdl32.dll. Replace other NIST MS Dll binaries according to Redistributables and Import Libraries table located in Technical Overview section of this document.
6. If your application is written in VB 6.0 or another different from C/C++ or compatible with the .NET Framework language consider switching to a .NET language and using NIST MS .NET interface assembly (NISTMSCL). Otherwise, carefully examine contents of NISTMS.H header file taking into account all preprocessor definitions used (you may find them in "Redistributables and Import Libraries" table or in CallDl32 project) and relevant to structure member alignment #pragma directives; make appropriate changes to your code.

How to update to version 2.1.3 *

If you are upgrading from version 2.0 or earlier, which does not recognize MS/MS search, make #define INTERNALBUILD visible in NISTMS.H.

How to update to version 2.0

If all data structures in your application passed to the DLL through nistms_search() are initiated with zeroes prior to initialized appropriate data members, the results should be nearly identical to those of the previous version.

The few differences in calling conventions are:

1. because of the inversion of the meaning of *alpha_only* parameter in Incremental Name Search, its values 0 and 1 should be interchanged;
2. the meaning of value *min_mass*=0 member of NISTMS_SRCH_CONTROLS data structure (compare peaks starting from the greatest out of two spectra lowest masses) has been changed; 0 should be replaced with –1, which means automatic selection of the lowest mass for spectral comparison.
3. The input spectrum for Chlorine - Bromine Estimation is pointed to by *userms*, not *libms*.
4. If your application may access user-created libraries containing chemical structures, it is recommended to call nistms_search() with *search_type* = NISTMS_INDEX_USER_STRU one time, right after your application starts and initiates the DLL, with all user libraries to be accessed listed as active. If structures in the user-created library are not properly indexed, adding a spectrum with chemical structure to the library or scanning structures in the library may fail.

If your application is written in C or C++ then only recompilation and relink is needed. If you use another programming language then you have to check and change your implementation of C structures listed at the end of section **"Summary of Implementation changes"** above as well as numerical values of *search_type* listed in the include file NISTMS.H.

However, in this way of porting the new features are not enabled except faster editing of user-created libraries.

Differences between v. 1.5 and v.2.0 user libraries

1. User libraries extensively edited with this DLL are not fully compatible with earlier versions of the DLL or with the NIST MS Search ver. 1.6d and earlier.
2. The current DLL can access and modify user libraries created with previous versions; however, it cannot access old NIST Main and Replicates libraries (released before 1998).
3. MS Search ver. 1.5x and earlier cannot access user libraries which contain structures.
4. MS Search ver. 1.6x and earlier may fail to access user libraries containing alternate chemical names (synonyms).

APPENDIX

1. Syntax of Name Fragments Constraint

char *io->constraints->name_frag[40]*
Applies to: all searches

This name constraint field can contain up to 8 name fragments.

The name fragments are searched in compound main chemical name and in other names (synonyms).

The following examples explain the conditions when the name constraint is satisfied (spaces around && and !! are optional):
"COCA && INE": at least one name contains both "COCA" and "INE"
"COCA && !! INE": at least one name contains "COCA" and doesn't contain "INE"
"COCA !! INE": at least one name contains "COCA"; no name contains "INE"

2. Syntax of Tags in Comment and Text Info Constraint

char *io->constraints->comment_tags[1024]*
Applies to: all searches

2.1. General

All information in NIST Peptide libraries is represented by the Tag=value convention. Typically, Peptide spectrum comment field may contain following items separated by spaces:

Tag=string
Tag="string"
Tag=number

Spaces may be inside a *string* only if the string is in double quotes. In older libraries, an unquoted *string* may also have space(s) inside a pair of parentheses ( ). Spaces are not permitted inside any *Tag* or next to the equal sign.

Comment fields in older libraries may also contain *Tag* not accompanied by the equal sign and *string*.

All *Tag*, *string*, *substring*, and *word* comparisons described below **ignore letter case**.

*2.1.1. Tags in Spectrum Text Information **

Text information displayed for NIST 11 MS/MS Database mass spectra has special tagged text fields, for example:

Instrument type: QqQ/triple quadrupole
Spectrum type: ms2
Compound type: M
Precursor type: [M+H]+
Precursor m/z: 223.0745
Collision energy: 6
Instrument: Micromass Quattro Micro
Sample inlet: direct flow injection
Ionization: ESI
Collision gas: Ar
Pressure: 1.6mTorr
Cone voltage: 25

Tags in Comment and Text Info Constraint allows searching these tagged fields.

2.2. Entering Tags in Comment Constraint

Items in the *Tags in Comment and Text Info* constraint field should be separated by spaces and/or linefeed characters. Unlike spectra comments, spaces next to equal sign (as well as >, <, >=, <=, and **:**) are allowed inside the items.

Spectra returned must satisfy all entered constraints

2.2.1 Examples

	Item	**Comments of the output spectra must have**
1	"*consensus*"	A Tag or a space-delimited string *consensus*
2	*parent*	6 letter sequence *parent* anywhere in the Comment
3	*fullname=”r.lecvk.c/2”*	Exactly this Tag=string:: *Fullname=R.LECVK.C/2*
4	*fullname=lec*	String following *Fullname=* contains sequence *LEC*
5	*Mods* **:** *ICAT_cl_hi*	String following *Mods=* contains word[1)] *ICAT_cl_hi*
	The following rules apply to strings that are numbers	
6	*Diffmod=6.8*	String following *Diffmod=* begins with number 6.8, for example, *Diffmod=6.8/8* or *Diffmod=6.8*
7	*Diffmod=5.1:6.8*	String following *Diffmod=* begins with number between 5.1 and 6.8, for example, *Diffmod=6.8/8* or *Diffmod=6.8*
8	*Diffmod > 6.8* *Diffmod >= 6.8* *Diffmod < 6.8* *Diffmod <= 6.8*	*Diffmod=* followed by a number greater than 6.8 *Diffmod=* followed by a number not less than 6.8 *Diffmod=* followed by a number less than 6.8 *Diffmod=* followed by a number not greater than 6.8
[1)] *word* is a string delimited by any character but a letter, a digit, or an underscore “_”; If *word* contains spaces or equal sign it should be entered in quotes.		

To reverse the meaning of constraints 1-5 (that is, change to “spectra must not have”), enter caret ^ as the first letter:

1	^"*consensus*"	Tag *consensus* is not present
2	^*parent*	Substring *parent* is not present
3	*fullname=*^"*r.lecvk.c/2*"	String tagged *fullname* is not exactly *r.lecvk.c/2*
4	*fullname=^lec*	String tagged *fullname* does not have substring *lec*
5	*Mods* **:** *^ICAT_cl_hi*	String tagged *Mods* has not word *ICAT_cl_hi*

2.2.2 More formal description

	Item	**Comments of the output spectra must have**
1	"*Tag*"	Exactly this *Tag* is present in the comment, possibly not followed by the equal sign
2	*substring*	this substring anywhere in the comment
3a	*Tag*="*string* "	exactly this *string* tagged with this *Tag*=
3b	*Tag*=""	This *Tag* has no *string* or its *string* is ""
4	*Tag*=*substring*	this *substring* in the *string* tagged with this *Tag*=
5a	*Tag* **:** *word*	exactly this *word inside string* tagged with this *Tag*=
5b	*Tag* **:** "*word*"	exactly this *word inside string* tagged with this *Tag*=
6	*Tag*=*number*	tagged string = *number* or *string* begins with *number*
7	*Tag*=*number***:***number*	tagged string *number* inside the given range
8a	*Tag* >= *number*	The number in the string less or equal to *number*
8b	*Tag* > *number*	The number in the string less then the *number*
8c	*Tag* <= *number*	The number in the string greater or equal to *number*
8d	*Tag* < *number*	The number in the string greater then the *number*

Notes.

1. Inside an item, spaces are allowed next to signs =, >, <, >=, <=, or **:**.. These signs as well as *Tag* and *string* or *number* or *word* may be located on separate lines.
2. If a *Tag* you created contains characters = or < or > or **:** then use double quotes to search for such a tag, for example, "<*Tag*>"="*string* ".
3. *substring* in rows 2 and 4 may not contain characters < > = **:**.
4. To search for a substring that may be interpreted as a number, for example, 1.23e5, use *Tag*==1.12e5 or *Tag*=="1.12e5" instead of *Tag*=1.12e5
5. To reverse the meaning of constraints 1-2 (that is, change to "spectra must not have"), insert caret ^ in front of the "*Tag*" or *substring*
6. To reverse the meaning of constraints 3-8 (that is, change to "spectra must not have"), insert caret ^ in front of the right hand side expression.
7. To use caret ^ as the first character of a *substring* enter *Tag* == "^*substring*".

Example of a multi-item constraint: The following items

eplicate	- Line 2, substring
RT >= 11.2	- Line 8a, number is greater or equal
Sample = "yeast_Nature_CAM"	- Line 3a, Tag="string"
Datfile = 011599.dat	- Line 4, Tag=substring

fit the following Comment field:

Replicate RT=11.220m Sample="yeast_Nature_CAM" Datfile="F011599.dat"

2.2.3 More formal description of reversed constraints

	Item	Comments of the output spectra must have
1a	^"*Tag*"	Exactly this *Tag* is not present in the comment
1b	~"*Tag*"	Some of the tags are not exactly this *Tag*
2a	^*substring*	this *substring* is not anywhere in the comment
2b	~*substring*	this *substring* is not anywhere in the comment
3a	^*Tag*="*string*"	all tags but this *Tag* have exactly this *string*
3b	~*Tag*="*string*"	some tags but not this *Tag* have exactly this *string*
3c	^*Tag*=^"*string*"	all tags but this *Tag* haven't exactly this *string*
3d	~*Tag*=^"*string*"	some tags but not this *Tag* haven't exactly this *string*
3e	*Tag*=^"*string*"	this *Tag* has no string or its string differs from *string*
3f	*Tag*=^""	this *Tag* has a non-empty string
3g	**all**=^"*string*"	Each of the tagged strings has not exactly this *string*
3h	**Some**="*string*"	some tagged strings have exactly this *string*

Items 4-8 from the previous section (1.2.2) may be given reversed meaning in the same way as 3a-3f.

Reserved expression **all=** is same as ^""= and ^=
Reserved expression **some=** is same as ~""= and ~=

To address a tag named **all** or **some** put it in quotes, for example:
"all"=*string* "some"=string
Quoting tags allows searching for exact combination of tags, for example
"Single Tryptic_simple Parent"=358.07 or
"Charge=2 Scan"=82

2.3. Entering Tags in Text Info Constraint *

Special tags are used for searching in the spectrum text information. To enter such a special tag, type dollar sign ($) immediately (no space) followed by the tag displayed in the spectrum text information, with spaces omitted or replaced with underscores. In the special tag, dashes, slashes, and underscores may be omitted, as well as # after letters "CAS". The search string and the special tag are case-insensitive. The syntax is almost the same as the one described above in the Entering Tags in Comment Constraint section. See also "Tagged Synonyms" section in this document.

*2.3.1. Examples **

To search for	Text Info Constraint Item
Precursor m/z in range[14] 102.9–103.2	$Precursor_m/z=102.9:103.2
Precursor m/z in range[14] 102.9–103.2	$precursormz=102.9:103.2
Instrument type containing word QqQ	$Instrument_type:QQQ
the exact Instrument name	$Instrument="Micromass quattro"
spectra that have any Related CAS# entry	$Related_CAS#

[14] Spectrum with text information "Precursor m/z: 103,77" also will be found because the first number, 103, is in the range 102.9–103.2. Note that the actual precursor m/z for such a ms^n spectrum (n > 2) is 77.

To search for	Text Info Constraint Item
spectra that have any Related CAS# entry	$relatedcas
spectra that have no Precursor m/z entry	^$Precursor_m/z

A quoted $:nn tag may be used instead of the special tag (see section "Tagged Synonym Display" for a list of $:nn tags). For example, "$:04"=102.9:103.2 may be used instead of $Precursor_m/z=102.9:103.2

*2.3.2. List of acceptable special tags **

$aux
$charge
$collision_energy
$collision_gas
$compound_type
$in-source_voltage
$instrument
$instrument_type
$ion_formula
$ion_mode
$ion_mw
$ion_name
$ionization
$known_impurity
$mass_range
$maximum_intensity
$peptide_mods
$peptide_sequence
$precursor_m/z
$precursor_type
$pressure
$related_cas#
$salt
$salt/mix_cas#
$sample_inlet
$special_fragmentation
$spectrum_type

3. Syntax of Peptide Sequence Constraint

char *io->constraints-> pep_name_frag[128]*
Applies to: Peptide spectra in Presearch=Off Peptide library search and in Sequential Method.

This constraint as well as other Peptide-specific constraints applies to Presearch=Off Peptide library search and Sequential Method (Other Searches) only. The constraint consists of one or more items separated by spaces, commas, or semicolons or located on different lines. If the first character of an item is caret (^) then the constraint is reversed: it is satisfied if the item is not present in the sequence. Depending on the character next to the caret or, in absence of caret, first character, an item matches peptide sequence either at a specified terminus or in any position inside a peptide chain.

First lowercase letter(s) or letter after the caret, ^	The item matches:
n	Sequence at N-terminus, starting from the leftmost amino acid
c	Sequence at C-terminus, ending at the rightmost amino acid
nc or cn	The whole sequence
Any other	Sequence starting in any position, from left to right

3.1. Alternatives to using dots in Peptide Sequence Constraint

.SEQUENCE is same as nSEQUENCE
SEQUENCE**.** is same as cSEQUENCE
.SEQUENCE**.** is same as ncSEQUENCE and cnSEQUENCE

Each item consists of units; each unit matches single amino acid. In the following table, A is single amino acid, S is a string containing one or more amino acid letters: S=A or $S=A_1A_2..A_n$

	Unit	Meaning
1	A	Matches one amino acid, A
2	[S]	Any of amino acids from S
3	[^S]	Any amino acid except those in S
4	?	Any amino acid
5	^A	Any amino acid but A, an abbreviation for [^A]

3.2. Examples (matched sequences are in **bold**):

Item	Matches
n??GEV	K.**ANGEV**VM(O)EM(O)NR.R/3
NS[EQ]N	EINAV**NSEN**K/2 and LQI**NSQN**CIHCK/3
cMNR	K.ANGEVVM(O)E**M(O)NR**.R/3
cMEM??	K.ANGEVV**M(O)EM(O)NR**.R/3
^[WYK]	Any sequence that does not contain any of W, Y, K
^WYK	The peptide does not contain WYK sequence

Note that

- [^WYK] means that at least one amino acid is not W or Y or K
- ^[WYK] means that any amino acid in a peptide is not W or Y or K.
- ^WYK means there is no WYK sequence in a peptide.

3.3. Abbreviations in Peptide Sequence Constraint

All single lowercase letters may be used to store parts or whole items. For example, "a=NS[E b=Q]N c=ab cMNR" is equivalent to "NS[EQ]NMNR". The first 3 items define "a" as NS[E, "b" as Q]N, and "c" as "a" and "b" concatenated into NS[EQ]N; the last item, cMNR, which is NS[EQ]NMNR, is the only item to be searched in peptide sequences. Since in this case "c" has been redefined; it does not mean C-terminus.

The items are interpreted from left to right and from top to bottom. Each abbreviation must be defined by the time it is being interpreted. Redefinitions are allowed, for example "a=NS[E b=Q]N ab a=[R ab" is equivalent to "NS[EQ]N [RQ]N".

Spaces next to the equal sign are not allowed and cause the rejection of the constraint.

Recursions are forbidden; for example "a=NSa a" will not be accepted.

Unused abbreviations, as "a=NS[E" in "a=NS[E cMNR" are ignored; this constraint is equivalent to "cMNR".

4. NISTMS_RECLOC type members of 2GB API

Data structure	**NISTMS_RECLOC type members**
NISTMS_AUX_DATA	spec_loc stru_loc *rep_locs

Data structure	NISTMS_RECLOC type members
NISTMS_HIT_LIST	*spec_locs *stru_pos
NISTMS_USER_STRUCT_INFO	*stru_pos
NISTMS_IO	input_spec_loc output_spec_loc

5. 2GB API NISTMS_RECLOC ftype values

The following table is mainly for debugging purposes. Typically, NISTMS_RECLOC::ftype should be considered a part of a spectrum or chemical structure location. Most of *ftype* values listed in the table will not appear in the NISTMS_RECLOC::ftype returned from NIST MS Dll. In future versions of NIST MS dll, the *ftype* values listed in the table may be subject to changes without notice.

ftype	Internal name	file names	Comment
0	COREDB	nist(.db, .inr), user.dbu	Main datafile
1	PEAKDB	peak(.db, .dbr, .dbu)	
2	LOSSDB	loss(.db, .dbr, .dbu)	
3	MAXMASSDB	maxmass(.db, .dbr, .dbu)	
4	SYNONYMDB	synonym.db	
5	CONTRIBDB	contrib(.db, .dbr)	
6	STRUCTDB	struct.db	Main struct. file
7	NAMEINCDB	namesort(.ask, .dbu)	
8	COREIN	nist(.in6, .inr), user.dbu	
9	STRUCTIN	struct.in6	
10	STCASIN	stcas.in6	
11	REGISTRYIN	registry(.in6, .inr, .inu)	CAS r.n. index
12	SPECNOIN	specno(.in6, .inr)	NIST r.n. index
13	NAMEIN	name.in6	
14	FORMIN	form(.in6, .inu)	
15	MWIN	mw(.in6, .inu)	
16	PEAKIN	peak(.in6, .inr, .inu)	
17	LOSSIN	loss(.in6, .inr, .inu)	
18	MAXMASSIN	maxmass(.in6, .inr, .inu)	
19	SYNONYMIN	synonym.in6	
20	CONTRIBIN	contrib(.in6, .inr)	
21	PEAKDB_AM2	peak_am2(.db, .dbr, .dbu)	
22	PEAKIN_AM2	peak_am2(.in6, .inr, .inu)	
23	PEAKDB_AM0	peak_am0(.db, .dbr, .dbu)	

ftype	Internal name	file names	Comment
24	PEAKIN_AM0	peak_am0(.in6, .inr, .inu)	
25	ALPHANAMEIN	alphanam(.in6, .inu)	
26	ALNUMNAMEIN	alnumnam(.in6, .inu)	
27	TEMPDB		
28	USRSTRUCTDB	usrstruc.db	Main struct. file
29	USRSTRUCRC	usrstcrc.inu	
30	USRSTRUPTR	usrstptr.dbu	
31	MZTEXT_DBU	mztxt.dbu	Exact m/z peaks file
32	MZTEXT_INU	mztxt.inu	
33	MZPRECUR_INU	precmz.inu	
34	PEAKDB_PM0	peak_pm0.dbu	
35	PEAKIN_PM0	peak_pm0.inu	
36	MZBIN_DBU	mzbin.dbu	Main MS/MS file
37	MZBIN_INU	mzbin.inu	
38	MZPRECUB_INU	precmzb.inu	
39	MWEXACTIN	exactmw(.in6, .inu)	Exact mass index
40	REGISTRYIN2	registry2(.in6, .inu)-	CAS r.n. index-2
41	PEAKDB_EM	peak_em(.db, .dbu)	Exact m/z peaks
42	PEAKIN_EM	peak_em(.in6, .inu)	Exact m/z peaks
50	USRSTRUCTTEMPDB	$nistms$.sdf	Temp. file
51	USRSTRUCTFILE	-	
52	USRSTRUSTDATA	-	

5. f32, a 32-bit Mass and m/z Floating Point Representation (version 2.1.5)*

Exact molecular mass in Da, accurate peak m/z and precursor m/z values are represented in a 32-bit unsigned floating point format with 5 bit exponent (bias=8) and 28 bit significand, including an implicit lead bit. Values in f32 format are saved as 32-bit unsigned integers.

Representation	Min. value	Value=0.75	Value=1.0	Max. value
Floating point	0.0039062500291038305	0.75	1.0	16,777,215.875
Hexadecimal int	0x00000001	0x3C000000	0x40000000	0xFFFFFFFE
int	1	1,006,632,960	1,073,741,824	4,294,967,294

Functions for converting double to f32 and back are exported from NIST MS Dll, starting from v.2.1.5.3. Their prototypes are available in the header file NISTMS.H:

```
double   nistms_f32_to_dbl( unsigned f32 );
unsigned nistms_dbl_to_f32( double d );
int      nistms_f32_dec_places( unsigned f32 );
```

The last function produces the number of decimal places to print the double value obtained from a value in f32 format to avoid trailing zeroes, for example:

```
unsigned f32  = 0x79507ae1; // f32 representation of 149.03
```

```
double   dVal = nistms_f32_to_dbl(f32 );
int      ndp  = nistms_f32_dec_places(f32 );
printf( "%.*f\n", ndp, dVal );
```