

Navigation and Ancillary Information Facility

Introduction to Kernels

September 2009



What is a SPICE “Kernel”

Navigation and Ancillary Information Facility

Kernel = File

Kernel = File containing ancillary data

Kernel = a file containing "low level" ancillary data that may be used, along with other data and SPICE software, to determine higher level observation geometry parameters of use to scientists and engineers in planning and carrying out space missions, and analyzing data returned from missions.



SPICE Kernels Family

Navigation and Ancillary Information Facility

- **SPK**
 - Spacecraft and Planet Ephemeris
- **Pck**
 - Planetary Constants, for natural bodies
 - » Orientation
 - » Size and shape
- **IK**
 - Instrument
- **CK**
 - Pointing (“C-matrix”)
- **EK**
 - Events, up to three distinct components
 - » ESP: science plan
 - » ESQ: sequence
 - » ENB: experimenter’s notebook
- **FK**
 - Reference frame specifications
- **SCLK**
 - Spacecraft clock correlation data
- **LSK**
 - Leapseconds
- **Meta-Kernel** (a.k.a. “FURNSH kernel”)
 - Mechanism for aggregating and easily loading a collection of kernel files
- **DSK (under development)**
 - Digital shape kernel
 - » Tessellated plate model
 - » Digital elevation model



Text and Binary Kernels

Navigation and Ancillary Information Facility

SPICE **text** kernels are:

- text PCK (the most common type of PCK)
- IK
- FK
- LSK
- SCLK
- MK (“Furnsh” meta-kernel)

SPICE **binary** kernels are:

- SPK
- binary PCK (exists only for Earth and moon)
- CK
- ESQ (part of the E-kernel)
- DBK (database kernel)
- DSK (digital shape kernel)*

* New kind of kernel under development



SPICE Kernel Forms

Navigation and Ancillary Information Facility

- **Binary form: SPK, binary PCK, CK, EK/ESQ¹, DSK**
 - Binary kernels are not human-readable and require the use of Toolkit software to examine the data contents.
- **Text form: text Pck, IK, FK, LSK, SCLK, FURNISH (MK)**
 - Text kernels contain only printing characters (ASCII values 32-126), i.e. human-readable text.
- **“Transfer” form of a binary kernel**
 - This is an ASCII representation of a binary kernel
 - Was used for porting the file between computers with incompatible binary representations (e.g. PC and UNIX)
 - Use of the transfer kernel is no longer needed for porting
 - » But is one way to convert a non-native binary kernel into native format, needed for modifying the kernel or improving read efficiency

[¹] The ESP and ENB components of the EK might be binary, text, or html, depending on specific implementation.



Example Text Kernel

Navigation and Ancillary Information Facility

This is a sample SPICE text kernel. The \begindata and \begintext markers on lines by themselves set off the start of data and text blocks respectively.

```
KPL/<kernel type>
\begindata
  NAME           = 'Sample text value'
  NaMe           = 'Keywords are case sensitive'

  NUMBERS        = ( 10.123, +151.241, -1D14 )
  NUMBERS        += ( 1.0,      1,      -10      )
  NUMBERS        += ( 1.542E-12, 1.123125412 )

  NAIF_BODY_NAME += ( 'SPEEDO', 'NEETO' )
  NAIF_BODY_CODE += ( -678,     -679     )

  TIME           = @1972-JAN-1

\begintext
  < some comments about the data >
\begindata
  < more data, given again in keyword = value syntax >
\begintext
  < etc., etc. >
```

The above assignments demonstrate that text kernels can contain characters, times, and numeric values. For more detailed information see Kernel Required Reading.



Text Kernel Operators

Navigation and Ancillary Information Facility

- The “+=” operator adds additional values to an existing variable. It creates a new variable if the referenced variable doesn’t already exist.
- The “@” symbol preceding a calendar date identifies a date string. The string must not contain embedded blanks. The string will be parsed and converted to an internal double precision representation of that epoch. The date is interpreted as ephemeris time (ET).
 - This conversion does not need a leapseconds kernel.

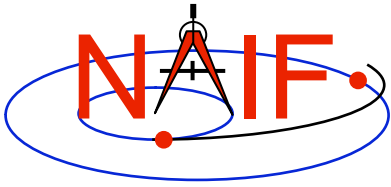


Example Binary Kernel

Navigation and Ancillary Information Facility

A binary kernel contains lots of non-printing (unintelligible) data, usually interspersed with occasional occurrences of ASCII characters.

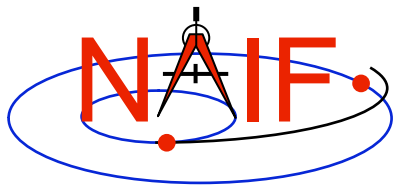
Other than moving binary kernels around on your computer, or between computers, the only way to use a binary kernel is to read it or add to it using a SPICE subroutine or program.



Metadata In SPICE Kernels

Navigation and Ancillary Information Facility

- **All SPICE kernels can and should contain metadata—descriptive information about the basic kernel data.**
- **Metadata might convey:**
 - when, where, how and by whom the file was made
 - for what purpose the file was made
 - what source data were used to make the file
 - key characteristics of the file, such as:
 - » start and stop times
 - » information about data gaps
 - » notes on expected data accuracy
 - cautions on use of the file
 - whatever else the file creator thinks could be useful to consumers of the data product



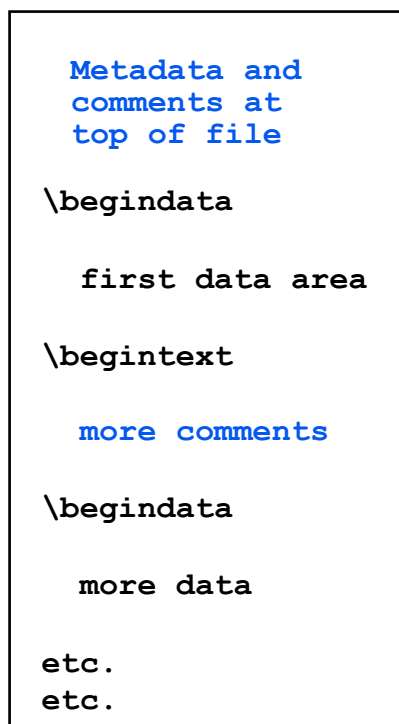
Examples of Metadata

Navigation and Ancillary Information Facility

SPICE Binary Kernel



SPICE Text Kernel



*Metadata in the comment area of a binary kernel can be accessed only using SPICE software.

Example of metadata as a PDS label

```
MISSION_NAME           = "MARS GLOBAL SURVEYOR"
SPACECRAFT_NAME        = "MARS GLOBAL SURVEYOR"
DATA_SET_ID            = "MGS-M-SPICE-6-V1.0"
KERNEL_TYPE_ID         = SPK
PRODUCT_ID             = "MGS_MAP1.BSP"
PRODUCT_CREATION_TIME  = 2000-01-14T08:26:18
PRODUCER_ID            = "MSOPNAV/JPL"
MISSION_PHASE_NAME     = MAPPING
PRODUCT_VERSION_TYPE   = ACTUAL
PLATFORM_OR_MOUNTING_NAME = "N/A"
START_TIME             = 1999-03-09T00:58:56
STOP_TIME              = 1999-06-02T00:58:56
etc.
```

Example of metadata as free-form text

Mars Global Surveyor Mapping SPK file, MGSNAV Solution

=====

Created by Boris Semenov, NAIF/JPL, June 14, 1999

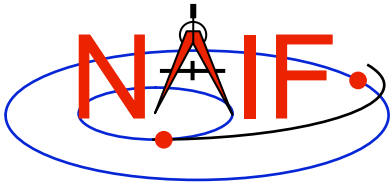
Objects in the Ephemeris

This file contains ephemeris data for the Mars Global Surveyor (MGS) spacecraft. NAIF ID code for MGS is -94.

Approximate Time Coverage

This file covers first three 28-day mapping cycles of the Mapping phase of the mission (mapping orbits 1 through 1040):

```
COVERAGE BEGIN TIME (TDB): 1999 MAR 09 01:00:00.000
COVERAGE END TIME (TDB): 1999 JUN 02 01:00:00.000
etc.
```



Loading Kernels - 1

Navigation and Ancillary Information Facility

- To make kernels available to SPICE programs you “load” them.
- When you load a text kernel:
 - the file is opened
 - the kernel contents are read into memory
 - » variable names and associated values are stored in a data structure called the “kernel pool”
 - the file is closed
- When you load a binary kernel:
 - the file is opened
 - for SPK, CK, and binary PCK files, no data are read until a read request is made by Toolkit software
 - for ESQ files, the schema description is read, checked, and stored in memory at load time, but no data are read until a query/fetch is made
 - for all practical purposes the binary file remains open unless specifically unloaded by you



Loading Kernels - 2

Navigation and Ancillary Information Facility

- Use the FURNISH routine to load all kernels—text and binary.
 - Sample FORTRAN, C, IDL and MATLAB calls:
 - » `CALL FURNISH ('name.ext')`
 - » `furnsh_c ("name.ext");`
 - » `cspice_furnsh, 'name.ext'`
 - » `cspice_furnsh ('name.ext')`
- **Best practices: don't hard code filenames—list these in a “meta-kernel” and load the meta-kernel using FURNISH. (See next page.)**
 - `CALL FURNISH ('meta-kernel_name')`
- **Caution: "Transfer format" versions of binary kernels can not be loaded; they must first be converted to binary with the Toolkit utility program *tobin* or *spacit*.**



What is a “Meta-Kernel”

Navigation and Ancillary Information Facility

- The terms “meta-kernel” and “FURNISH kernel” are used synonymously
- Using a meta-kernel makes it easy to manage which SPICE files are loaded into your program
- A meta-kernel is a file that lists names (and locations) of a collection of SPICE kernels that are to be used together in some SPICE-based application
 - You can then simply load the meta-kernel, causing all of the kernels listed in it to be loaded
- A meta-kernel is implemented using the SPICE text kernel standards



Sample Meta-Kernel Contents (1)

Navigation and Ancillary Information Facility

- This is a sample meta-kernel that the Toolkit routine FURNISH could use to load a collection of kernels.

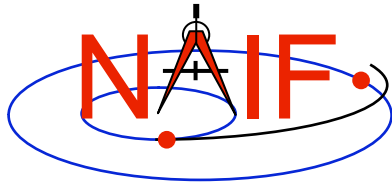
KPL/MK

\begindata

```
KERNELS_TO_LOAD = (  
    '/home/mydir/kernels/lowest_priority.bsp',  
    '/home/mydir/kernels/next_priority.bsp',  
    '/home/mydir/kernels/highest_priority.bsp',  
    '/home/mydir/kernels/leapseconds.tls',  
    '/home/mydir/kernels/sclk.tsc',  
    '/home/mydir/kernels/c-kernel.bc',  
    '/home/mydir/kernels+',  
    '/custom/kernel_data/p_constants.tpc',  
)
```

The commas
are optional

- The last file listed in this example (p_constants.tpc) demonstrates how to use the continuation character '+' to work around the 80 character limitation imposed on string sizes by the text kernel standards.



Sample Meta-Kernel Contents (2)

Navigation and Ancillary Information Facility

- This sample meta-kernel uses the **PATH_VALUES** and **PATH_SYMBOLS** keywords to specify the directory where the kernels are located.

KPL/MK

\begindata

```
PATH_VALUES      = ( '/home/mydir/kernels' )
PATH_SYMBOLS     = ( 'KERNELS' )
KERNELS_TO_LOAD = (
    '$KERNELS/lowest_priority.bsp',
    '$KERNELS/next_priority.bsp',
    '$KERNELS/highest_priority.bsp',
    '$KERNELS/leapseconds.tls',
    '$KERNELS/sclk.tsc',
    '$KERNELS/c-kernel.bc',
    '$KERNELS/custom/kernel_data/p_constants.tpc'
)
```

- Although the OS environment variable notation **\$NAME** is used to refer to the symbols set by the **PATH_VALUES** and **PATH_SYMBOLS** keywords, these symbols are **NOT** operating system environment variables and are set and used for substitution by **SPICE** only in the context of this particular meta-kernel.
- The '+' continuation character described on the previous page may also be used to handle path values strings that exceed 80 characters.



Limits on Loaded Kernels

Navigation and Ancillary Information Facility

- **The number of binary kernels that may be loaded at any time is large, but limited.**
 - For SPK, CK, and binary PCK files:
 - » Loaded SPKs + Loaded CKs + Loaded binary PCKs ≤ 1000
 - For ESQ files:
 - » Loaded ESQs ≤ 20
 - For all kernels:
 - » Loaded kernels ≤ 1300
 - Assumes each has been loaded only once, and not unloaded.
- **There are also limits on the number of keywords and values for all loaded text kernels:**
 - Maximum number of keywords is 5003.
 - Maximum number of numeric data items is 40,000.
 - Maximum number of character data items is 4000.



Kernel Precedence Rule

Navigation and Ancillary Information Facility

- **The order in which SPICE kernels are loaded at run-time determines their priority when requests for data are made**
 - For binary kernels, data from a higher priority file will be used in the case when two or more files contain data overlapping in time for a given object.
 - » For SPKs, CKs and binary PCKs the file loaded **last** takes precedence (has higher priority).
 - » Priority doesn't apply to ESQ files – all data from all loaded files are available.
 - If two (or more) text kernels assign value(s) to a single keyword using the “=” operator, the data value(s) associated with the last loaded occurrence of the keyword are used—all earlier values have been replaced with the last loaded value(s).
 - Orientation data from a binary PCK always supersedes orientation data (for the same object) obtained from a text PCK, no matter the order in which the kernels are loaded.



Unloading Kernels

Navigation and Ancillary Information Facility

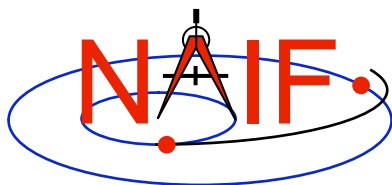
- **The unloading of a kernel is infrequently needed for FORTRAN or CSPICE applications but is essential for Icy and Mice scripts**
 - Because of the way IDL and MATLAB interact with external shared object libraries any kernels loaded during an IDL or MATLAB session will stay loaded until the end of the session unless they are specifically unloaded
- **The routines KCLEAR and UNLOAD may be used to unload kernels containing data you wish to be no longer available to your program.**
 - KCLEAR unloads all kernels and clears the kernel pool
 - UNLOAD unloads specified kernels
 - KCLEAR and UNLOAD are only capable of unloading kernels that have been loaded with the routine FURNISH. They will not unload any files that have been loaded with older load routines such as SPKLEF (those used prior to availability of FURNISH).
- **Caution: unloading text kernels with UNLOAD will also remove any kernel pool data provided through the kernel pool API (P*POOL)**



Backup

Navigation and Ancillary Information Facility

- **How kernels are made and used**
- **Why and how kernels are modified**
- **SPICE data structures hierarchy**

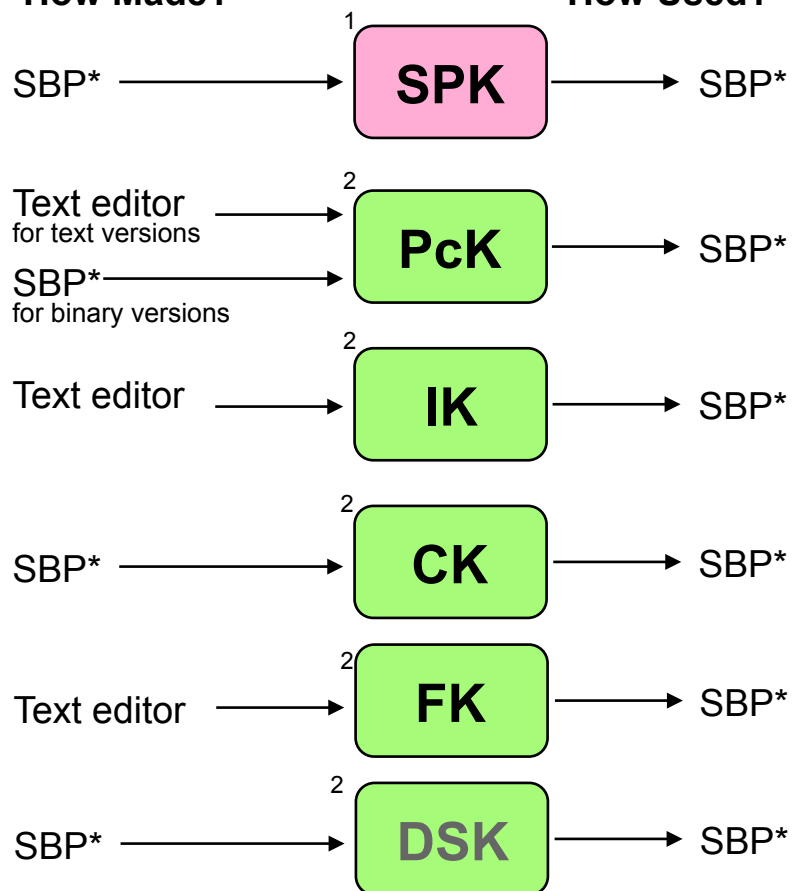


How Kernels are Made and Used at JPL

Navigation and Ancillary Information Facility

How Made?

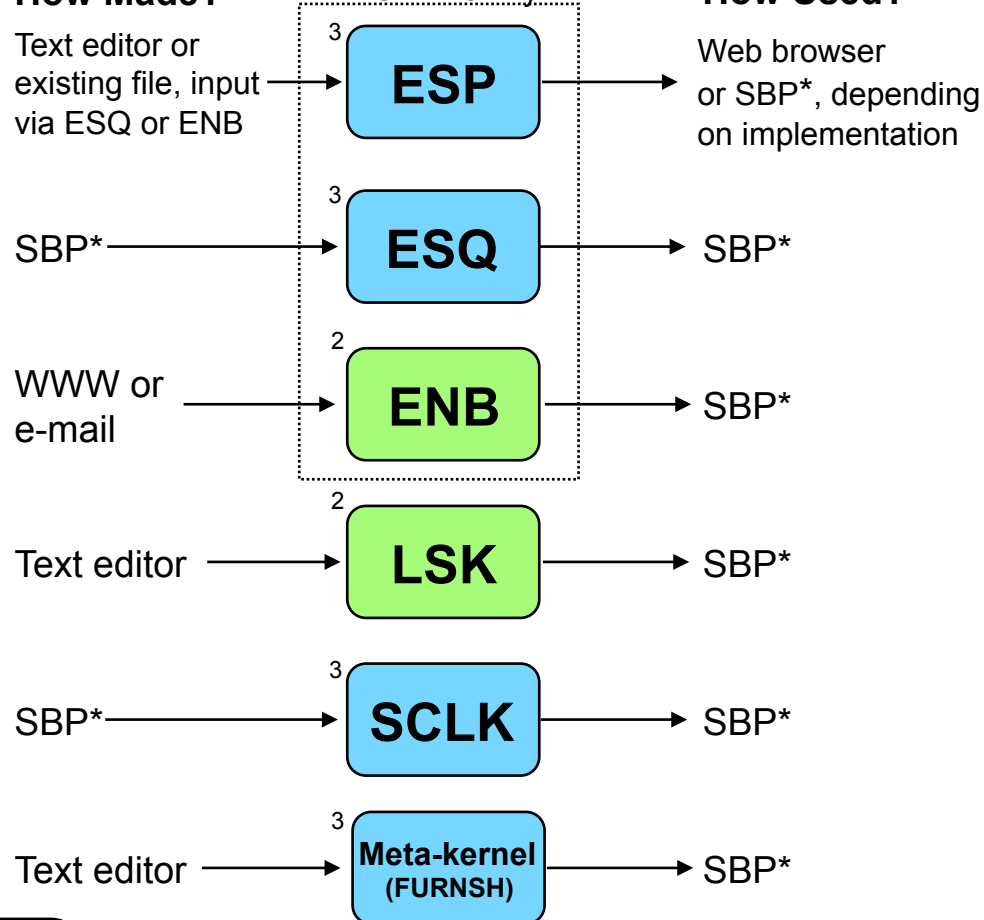
How Used?



How Made?

The EK family

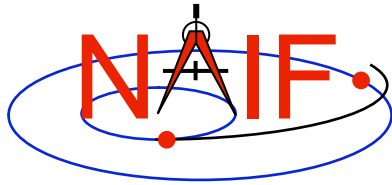
How Used?



Who usually makes the kernels at JPL?

- 1 (pink circle) **NAV and NAIF** This represents current practice for most JPL missions, but is by no means a requirement. **Anyone can make SPICE files.**
- 2 (green circle) **NAIF**
- 3 (blue circle) **NAIF or other**

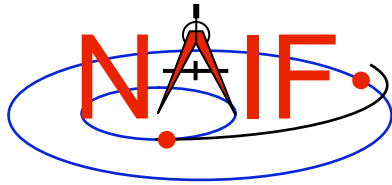
***SBP** = SPICE-based program that uses modules from the SPICE Toolkit. In some cases the Toolkit contains such a program already built. In some cases NAIF may have such a ready-built program that is not in the SPICE Toolkit.



Why & How Kernels are “Modified” - 1

Navigation and Ancillary Information Facility

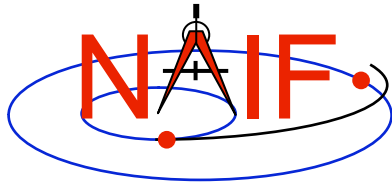
<u>File Type</u>	<u>Why Modified</u>	<u>How Modified</u>
SPK	<ul style="list-style-type: none"> -To add metadata (comments) -To merge files or subset a file -To correct/revise an object ID 	<ul style="list-style-type: none"> - COMMNT, SPACIT or SPICELIB module - SPKMERGE - BSPIDMOD
PcK Text version	<ul style="list-style-type: none"> -To revise data values -To add additional data items and values 	<ul style="list-style-type: none"> - Text editor - Text editor
IK	<ul style="list-style-type: none"> -To revise data values -To add additional data items and values 	<ul style="list-style-type: none"> - Text editor - Text editor
CK	<ul style="list-style-type: none"> -To add metadata (comments) -To merge files -To revise the interpolation interval -To subset a file 	<ul style="list-style-type: none"> - COMMNT, SPACIT, or SPICELIB module - DAFCAT, CKSMRG - CKSPANIT, CKSMRG - CKSLICER
FK	<ul style="list-style-type: none"> -To revise data values -To add additional data items and values 	<ul style="list-style-type: none"> - Text editor - Text editor
DSK	<ul style="list-style-type: none"> -To add metadata (comments) -To merge files or subset a file 	<ul style="list-style-type: none"> - COMMNT, SPACIT or SPICELIB module - DSKMERGE



Why & How Kernels are “Modified” - 2

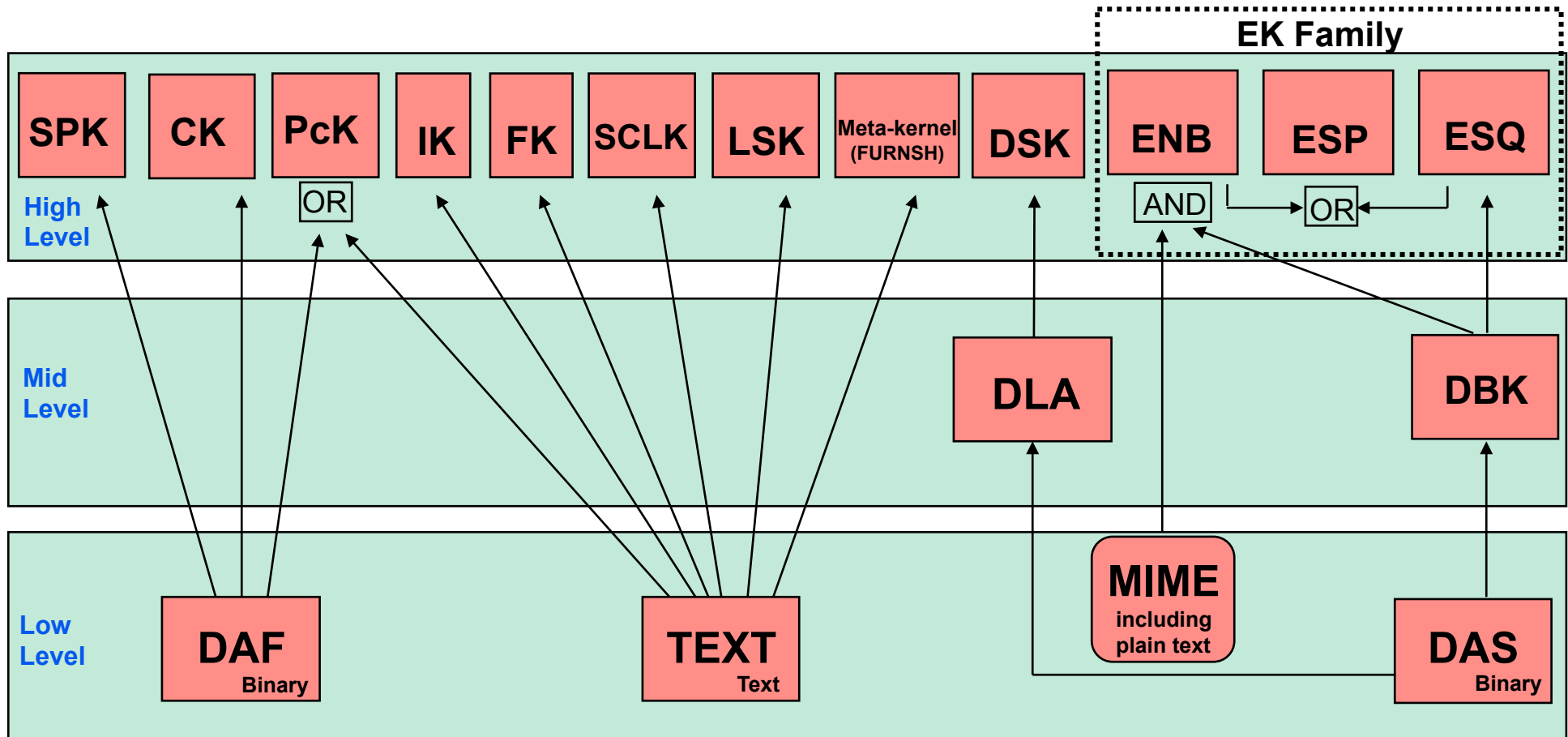
Navigation and Ancillary Information Facility

<u>File Type</u>	<u>Why Modified</u>	<u>How Modified</u>
The EK family		
ESP	<ul style="list-style-type: none"> -To add, revise or delete “data” -To add metadata (comments) 	<ul style="list-style-type: none"> - (Depends on implementation) - (Depends on implementation)
ESQ	<ul style="list-style-type: none"> -To add additional data -To revise data -To delete data -To add metadata (comments) -To merge files 	<ul style="list-style-type: none"> - Toolkit modules - Toolkit modules - Toolkit modules - COMMNT, SPACIT or SPICELIB module - (under development)
ENB	<ul style="list-style-type: none"> -To change entry status (public <--> private) -To delete an entry 	<ul style="list-style-type: none"> - WWW - WWW
LSK	<ul style="list-style-type: none"> - To add a new leapsecond 	<ul style="list-style-type: none"> - Text editor
SCLK	<ul style="list-style-type: none"> - To add metadata 	<ul style="list-style-type: none"> - Text editor
Meta-kernel (FURNISH)	<ul style="list-style-type: none"> - To revise contents in any way 	<ul style="list-style-type: none"> - Text editor



SPICE Data Structures Hierarchy

Navigation and Ancillary Information Facility



DAF = Double Precision Array File

DSK = Digital Shape Kernel (under development)

DBK = Data Base Kernel

DLA = DAS Linked Array (under development)

DAS = Direct Access, Segregated

Excepting MIME, each of these data structures is built entirely of SPICE components.

PcK files are usually text-based, but binary versions exist for the earth and moon. The ESP has been implemented using both the ENB and ESQ mechanisms. The DBK is a SQL-like, homebrew database.