# MAVEN IDL Toolkit Users’ Guide

V20170512 (2017-May-12)

## Table of Contents

1. Toolkit Installation
   1.1. System Requirements ................................................................. 2
   1.2. Downloading the Toolkit ............................................................ 2
   1.3. Updating the Toolkit
       1.3.1. Automatic Updater .............................................................. 3
       1.3.2. Manual Update ................................................................. 3
   1.4. Installing the Toolkit ................................................................. 3
   1.5. IDL Configuration .................................................................. 4
   1.6. Testing Your Toolkit Installation .................................................. 5
   1.7. Required Data Directory Structure ............................................... 5
   1.8. Getting More Help .................................................................. 6

2. Toolkit Routines
   2.1. Downloading Data Using the Toolkit ............................................. 7
       2.1.1. mvn_kp_download_files ...................................................... 7
       2.1.2. mvn_kp_download_l2_files ................................................ 8
       2.1.3. mvn_kp_download_orbit_file .............................................. 9
   2.2. Reading data into IDL Memory ...................................................... 10
       2.2.1. mvn_kp_read ................................................................. 10
       2.2.2. mvn_kp_read_model_results ............................................. 13
   2.3. Manipulating Key Parameter Data ............................................... 14
       2.3.1. mvn_kp_insitu_search ....................................................... 14
       2.3.2. mvn_kp_iuvs_search ....................................................... 16
       2.3.3. mvn_kp_bin ................................................................. 17
       2.3.4. mvn_kp_resample ......................................................... 19
   2.4. Plotting Key Parameter Data ....................................................... 20
       2.4.1. mvn_kp_plot ................................................................. 20
       2.4.2. mvn_kp_altplot ............................................................ 23
       2.4.3. mvn_kp_standards .......................................................... 23
       2.4.4. mvn_kp_tplot .............................................................. 26
       2.4.5. mvn_kp_iuvs_limb ........................................................ 28
       2.4.6. mvn_kp_iuvs_corona .................................................... 30
   2.5. Interpolating Model Results ....................................................... 32
       2.5.1. mvn_kp_interpol_model .................................................. 32
       2.5.2. mvn_kp_create_model_maps ........................................... 33
   2.6. Adding User-supplied Data ....................................................... 33
       2.6.1. mvn_kp_add_data .......................................................... 33
   2.7. Plotting of Key parameter data in 2D ......................................... 35
       2.7.1. mvn_kp_map2d ............................................................. 35
   2.8. Visualization of 3D data ........................................................... 37
       2.8.1. mvn_kp_3d ................................................................. 37
   2.9. Checking for Toolkit Updates .................................................... 41
       2.9.1. mvn_kp_check_version ................................................... 41
1. Toolkit Installation

1.1. System Requirements

The MAVEN Toolkit currently requires IDL version 8.1 or above. We recommend IDL 8.3, as this has been the most stable version in all of our testing. The Toolkit should run equally well on Windows, or UNIX-like (Linux, Mac OS-X, etc.) systems, though development took place entirely within Mac OS-X so there might be small differences between operating systems.

1.2. Downloading the Toolkit

The MAVEN Toolkit is available for download at the MAVEN Science Data Center team website.

https://lasp.colorado.edu/maven/sdc/team/pages/software.html

Or, for non-MAVEN Science Team members, it will be available for download at the MAVEN mission public site (below). The only difference between the two is the time window of available data, in accordance with the PDS and NASA data sharing requirements:

https://lasp.colorado.edu/maven/sdc/public/pages/software.html

The file you are looking for will be named **Toolkit_V#.##.zip**, depending upon the version number of the current release. As of 12 May 2017, the current available version is **V20170512**.

You will also need to download Berkeley’s tplot/TDAS ssl_general code. This can be obtained from a link to Berkeley’s servers that will be provided at the MAVEN Science Data Center team website.

Because each package is undergoing constant development, conflicts can occur if you run software that is not up to date. New releases of the MAVEN Toolkit will be tested to ensure compatibility with the most current version of the Berkeley tplot/TDAS software. *If you encounter unexpected or previously unseen errors after installing the MAVEN Toolkit, first ensure that you also have downloaded the most current version of the Berkeley tplot/TDAS software.*
1.3. Updating the Toolkit

1.3.1. Automatic Updater

In order to make sure the user has access to the latest code, the procedure `mvn_kp_check_version` (Section 2.9) is run periodically to check for updates to the toolkit. You may occasionally see a prompt on the command line that looks like:

“There is a new version of the software. Would you like to download it now (y/n)? :”

If you select “y”, the latest files will be installed and the toolkit will continue on to the previous command. If you select “n”, the toolkit will not ask you again until IDL is reset.

It is highly recommended that you download the latest version whenever this prompt appears. The Key Parameter data format changes often, and it is likely that updates to the toolkit were made to accommodate these changes.

If you would like to see a description of the latest changes before downloading, you can find a version history on the MAVEN SDC website at [https://lasp.colorado.edu/maven/sdc/public/data/sdc/software/idl_toolkit/Source/Version_History.txt](https://lasp.colorado.edu/maven/sdc/public/data/sdc/software/idl_toolkit/Source/Version_History.txt)

You can also check for new versions at any time by simply running the procedure `mvn_kp_check_version`

1.3.2. Manual Update

If you ever need to manually update the toolkit, simply download the latest version of the toolkit from the team or public website, and follow the steps in the next section.

1.4. Installing the Toolkit

Skip this first paragraph if this is your first installation of the toolkit, otherwise the following describes how to install an updated version of the toolkit:

1. Copy your `mvn_toolkit_prefs.txt` file somewhere other than your installation directory.
2. If your data files are located in your installation directory, move your data files elsewhere.
3. Remove all contents of previous toolkit installation.
4. After you’ve unzipped the new toolkit into your installation directory, move the `mvn_toolkit_prefs.txt`, and data files if applicable, back into your installation directory.
For new users and previous users: The MAVEN Toolkit may be installed in any directory on your computer. Simply extract the downloaded file to the location of your choice, and the Toolkit will create the necessary subdirectories. Recall the path to where you’ve extracted the Toolkit, this path will be needed in the next step.

1.5. IDL Configuration

IDL will need to be configured to know where to find both the MAVEN Toolkit and tplot/TDAS routines. This is done by setting the IDL Path variable.

If you use IDL from the command line:
You need to set the IDL_PATH environment variable such that IDL knows where to find the Toolkit (and tplot/TDAS routines). Doing this depends on your system.

(N.B., below: +path/to/Toolkit is the actual path to where you’ve unzipped the toolkit. And, the + symbol is necessary)

1. If you use csh or tcsh (MAC OSX), add this line to the end of your ~/.cshrc file:

```plaintext
setenv IDL_PATH '<IDL_DEFAULT>:+path/to/Toolkit'
```

Note: If you already set the IDL_PATH somewhere, then you'll want to append the :+path/to/Toolkit portion of the above example to your existing setenv of IDL_PATH

2. If you use bash or sh (Linux or some Macs) then add this line to the end of your ~/.bashrc file:

```plaintext
export IDL_PATH='<IDL_DEFAULT>:+/path/to/Toolkit'
```

Note: If you already set the IDL_PATH somewhere, then you'll want to append the :+path/to/Toolkit portion of the above example to your existing setting of IDL_PATH.

For example, on MAC OSX I added the following line to the end of my ~/.bashrc file (I use bash):

```plaintext
export IDL_PATH='<IDL_DEFAULT>:+/Users/martin/maven_toolkit/'
```

3. If you use the IDL Developers Environment:
   a. Open the preferences window:
      i. IDL... Preferences (on a Mac or *nix machine)
      ii. Window... Preferences (on a Windows machine)
   b. From the left pane, expand the IDL options and select “Paths.”
   c. Select “Insert.”
d. Navigate to the directory in which the Toolkit is installed and click “Open.”
e. Select the new path that has just been added to the list and click the checkbox (directly to the left of the path). This is required so that all subdirectories will be added to the PATH.

**NOTE:** We’re assuming you’ve set up your IDL path to find your installation of tplot/ssl_general. If you’re unclear, refer to documentation at [http://themis.ssl.berkeley.edu/software.shtml](http://themis.ssl.berkeley.edu/software.shtml).

### 1.6. Testing Your Toolkit Installation

To check if your IDL path configuration is properly set up to find the Toolkit and tplot routines, type the following commands (if using terminal, make sure to restart it after adding path):

```
IDL> libs, 'mvn_kp_read'
IDL> libs, 'mvn_kp_test_install'
IDL> libs, 'tplot'
```

If IDL can find the necessary routines, you will see the full path to the `mvn_kp_read`, `mvn_kp_test_install`, and `tplot` routines. If there is more than one entry for a routine then you have multiple copies of either the Toolkit or SSL/tplot routines installed on your computer and there might be conflicts, as IDL will find multiple versions to run.

To check that a couple of basic KP file reads work, and to see if you need the IDL CDF patch, type the following command:

```
IDL> mvn_kp_test_install
```

If the output prints two “SUCCESS” lines, you are done! If you receive any other type of error, email us for help.

### 1.7. Required Data Directory Structure

The toolkit requires data files to be stored in a particular directory structure. This directory structure matches the SDC and SSL directory structure. However, a user is able to choose the root directory location for the data to be stored (will refer to this as the **ROOT_DATA_DIR**). When you first use a download or read procedure, you will be prompted to select this **ROOT_DATA_DIR**. After the selection is made, it is saved in a file (mvn_toolkit_prefs.txt), and can be changed later if desired. A user can also choose to set an environment variable **ROOT_DATA_DIR** instead of using the preferences file. The remaining directory structure will then be automatically generated under this **ROOT_DATA_DIR**. The toolkit download procedures will download files into this directory structure, and the read procedure expects to find
data files there as well. The directory structure that is created (and required) looks like:

```
<ROOT_DATA_DIR>/maven/data/sci/kp/insitu/<YYYY>/<MM>/
    kp/iuvs/<YYYY>/<MM>/
```

And if you choose to download the level 2 data using the Toolkit routine (mvn_kp_download_l2_files), the following will be created (and required):

```
<ROOT_DATA_DIR>/maven/data/sci/sta/l2/<YYYY>/<MM>/
    sep/l2/<YYYY>/<MM>/
    swi/l2/<YYYY>/<MM>/
    swe/l2/<YYYY>/<MM>/
    lpw/l2/<YYYY>/<MM>/
    mag/l2/<YYYY>/<MM>/
    iuv/l2/<YYYY>/<MM>/
    ngi/l2/<YYYY>/<MM>/
    euv/l2/<YYYY>/<MM>/
    acc/l2/<YYYY>/<MM>/
```

- **Note:** If you are on Windows, the forward slashes (/) will be back slashes (\).
- `<ROOT_DATA_DIR>` is chosen by the user.
- `<YYYY>` and `<MM>` will be created as files for those year/month exist.

### 1.8. Getting More Help

If you have any further problems or questions, either with installation or operation of the Toolkit, feel free to contact the developers at maven_divide@lasp.colorado.edu

### 2. Toolkit Routines

Here we list the routines available in the MAVEN Toolkit and provide static documentation for their usage. For each, we will list the procedure name, followed by a brief description of what the code does, including the required and optional arguments that can be passed to the procedure. Next, we display an example usage of the procedure, followed by a comprehensive listing of all of the required arguments, then the optional arguments.

For the reader unfamiliar with IDL, valued optional arguments must be passed using the syntax “argument=value” while flags or Boolean arguments are passed using the syntax “/flag”. The ordering of required arguments must be preserved if data are being passed to or from the procedure; otherwise, arguments may be listed in any order.

For all of these procedures, issuing the command “<procedure_name>, /help” will generate a brief listing of the information contained here.
2.1. Downloading Data Using the Toolkit

2.1.1. mvn_kp_download_files

Description
This procedure downloads in-situ and/or IUVS Key Parameter (KP) data files from the MAVEN SDC web server.

Example Usage
- Download all available KP data files having CDF format for the in situ instruments between 1 January 2015 and 31 January 2015, inclusive:

  IDL> mvn_kp_download_files, start_date='2015-01-01', end_date='2015-01-31', /insitu, /cdf_files

- List all available CDF in-situ KP files on the server:

  IDL> mvn_kp_download_files, /insitu, /cdf_files, /list_files

- Download all new (those files on the server that are not found in your local data directory) text (ASCII) IUVS KP files through 6 April 2015:

  IDL> mvn_kp_download_files, /iuvs, /cdf_files, /new_files, end_date='2015-04-06'

Required Arguments
Either /insitu or /iuvs must be specified; but not both (at the present time). At least one of /text_files or /cdf_files must be specified.

List of all accepted Arguments
- /insitu: Search/download in-situ KP data files
- /iuvs: Search/download IUVS KP data files
- /cdf_files: Search/download CDF formatted data files (*.cdf)
- /text_files: Search/download ASCII formatted data files (*.tab)
- /new_files: Search/download only files that exist on the server
- /list_files: I.e., “dry run.” List to standard output (usually, the screen) the files that would be downloaded based on the provided arguments; but do not download any data.
- /update_prefs: Before searching/downloading data, open up a dialogue window to allow the user to update the mvn_toolkitPrefs.txt file containing the location of the ROOT_DATA_DIR. Once the new path is selected, the search/download will proceed according to the remaining arguments.
- /only_update_prefs: As /update_prefs; but do not attempt to search download and data.
• /exclude_orbit_file: Do not download an updated version of the orbit number file from http://naif.jpl.nasa.gov/naif/.
• filenames: scalar or an array of specific filename strings to search/download. The full filename must be provided (e.g., mvn_kp_insitu_20150129_v00_r01.tab); wildcards are not recognized.
• start_date: (format='YYYY-MM-DD') Search/download only data from start_date (inclusive) to present.
• end_date: (format='YYYY-MM-DD') Search/download only data from prior to end_date (inclusive).
• local_dir: Specify a directory to which to download files. This overrides (but does not overwrite) the target listed in the mvn_toolkit_prefs.txt file.
• /debug: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
• /help: Invoke this list.

2.1.2. mvn_kp_download_l2_files

Description
This procedure downloads Level 2 data files for any instrument from the MAVEN SDC web server. Upon the first invocation of this procedure, the required directory structure will be built up at the location of your ROOT_DATA_DIR, defined in your mvn_toolkit_prefs.txt file.

Example Usage
– List to screen all available Level 2 data files for the SWIA instrument.

IDL> mvn_kp_download_l2_files, instrument='swi', /list_files

– List all available Level 2 data files for the SWIA instrument for the month of January, 2015.

IDL> mvn_kp_download_l2_files, instrument='swi', /list_files, start_date='2015-01-01', end_date='2015-01-31'

– Download all available Level 2 data files for the NGIMS, STATIC, and EUV instruments that currently exist at the SDC server, but not in your local data directory.

IDL> mvn_kp_download_l2_files, instruments=['ngi','sta','euv'], /new_files

Required Arguments
At least one instrument must be provided.
List of all accepted Arguments

- **instruments**: Scalar or an array of 3-character string abbreviations (the instrument-specific directory names in the directory structure) of instruments for which the data are to be downloaded/searched.
- **/new_files**: Search/download only files that exist on the server
- **/list_files**: I.e., “dry run.” List to standard output (usually, the screen) the files that would be downloaded based on the provided arguments; but do not download any data.
- **/update_prefs**: Before searching/downloading data, open up a dialogue window to allow the user to update the mvn_toolkit_prefs.txt file containing the location of the ROOT_DATA_DIR. Once the new path is selected, the search/download will proceed according to the remaining arguments.
- **/only_update_prefs**: As /update_prefs; but do not attempt to search download and data.
- **/exclude_orbit_file**: Do not download an updated version of the orbit number file from http://naif.jpl.nasa.gov/naif/.
- **filenames**: scalar or an array of specific filename strings to search/download
- **start_date**: (format='YYYY-MM-DD') Search/download only data from start_date (inclusive) to present.
- **end_date**: (format='YYYY-MM-DD') Search/download only data from prior to end_date (inclusive).
- **/debug**: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
- **/help**: Invoke this list.

2.1.3. mvn_kp_download_orbit_file

**Description**

This will download an orbit number file from JPL into the toolkit installation directory. If the orbit file already exists, this will overwrite it with the most up-to-date orbit number file. This file is necessary if one wants to read in with orbit number for time range.

**Example Usage**

```
IDL> mvn_kp_download_orbit_file
```

**Required Arguments**

None

**List of all accepted Arguments**

- **/debug**: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
2.2. Reading data into IDL Memory

2.2.1. mvn_kp_read

Description

This procedure ingests a subset of locally available KP data into one or two data structures in IDL memory (depending on the provided arguments). The data structure variables thus produced are the primary inputs to the various plotting routines contained within this Toolkit.

In its simplest form, the read routine will quickly and efficiently return all MAVEN KP data with a single command. Through the use of a variety of keywords and options, however, the user may use the same routine to extract any subset of KP data according to their needs.

The first time a user calls `mvn_kp_read`, a dialog will appear that ask for the location for `ROOT_DATA_DIR`, where on their machine MAVEN KP data is stored (see Required Directory Structure). If a user has already run a download routine, then this selection has already been made and saved.

After the first time, the Toolkit routines will remember this location and not prompt the user again. However, should the user wish to change this, or access KP data in an alternate directory, they can re-enter this dialog via optional keywords (below).

Example Usage

- Read one day’s worth (10 April 2015) of in-situ and IUVS KP data

  `IDL> mvn_kp_read, ‘2015-04-10’, insitu, iuvs`

- Read in-situ and IUVS KP data from a given start time (1:05:01 on 5 April 2015) to a given end time (14:22:11 on 22 April 2015). NB, the procedure also accepts time/date strings in the PDS convention: ‘yyyy-mm-ddThh:mm:ss’.


- Read one orbit (orbit 501) of only in-situ data

  `IDL> mvn_kp_read, 501, insitu, /insitu_only`

- Read one day (1 April 2015) of Magnetometer, NGIMS, and IUVS Coronal Echelle data from the inbound leg of the orbit. N.B., since IDL recognizes keywords the moment they cease to be ambiguous, if you use the three-letter abbreviations for the in-situ instruments, they will be recognized (e.g., “/ngi” for “/ngims” below).
Read in five days (19 April 2015 to 24 April 2015) of in-situ and IUVS data, downloading any new files from the SDC server that are not already stored locally.

IDL> `mvn_kp_read`, '2015-04-01', insitu, iuvs, /mag, /ngi, /iuvs_coronaEchelleHigh, /iuvsCoronaEchelleLimb, /iuvs_coronaEchelleDisk, /inbound

```
IDL> `mvn_kp_read`, ['2015-04-19/00:00:00', '2015-04-24/00:00:00'], insitu, iuvs, /download_new
```

**Required Arguments**

**time:**
The user is required to provide a constraint on the window for which data is to be retrieved from the local disk and stored in the local IDL data structure(s). These constraints may be provided in one of three formats:

1. As a date string (format='YYYY-MM-DD')
2. As a date/time string (format='YYYY-MM-DD/HH:MM:SS')
   - (also, format = 'yyyy-mm-ddThh:mm:ss')
3. As an orbit number

At a minimum, the user is required to input a single time from which the KP data will be read. If the user inputs a single time, then the routine will read data for the default time period, beginning at the entered time. Alternately, the user may enter time as a two-element array that corresponds to the beginning and end times to be read. **N.B., the default read period, if only a single time or orbit is entered by the user, is defined a single day (86400 seconds) if a date and time is entered, or a single orbit if orbit is passed.**

**insitu_output:**
This user-defined variable will be the name of the structure returned that contains all the in-situ instrument KP data as well as the spacecraft position and orientation information. The INSITU_OUTPUT structure is always filled with some data, even if only IUVS data is requested via keyword because it also contains the spacecraft position and orientation information that is needed for later visualization.

**iuvs_output:**
This user-defined variable will be the name of the structure returned that contains all the IUVS KP data.

**List of all accepted Arguments**

- **time:** Must be the **first** argument after the procedure call.
  - Description above in “Required Arguments”
- **insitu_output:** Must be the **second** argument after the procedure call.
  - Description above in “Required Arguments.”
- **iuvs_output:** Must be **third** argument after the procedure call
  - Only required if `/insitu_only` keyword is **not** set
- Description above in “Required Arguments.”

- `/insitu_only`: Setting this keyword will read in only the KP data from the in-situ instruments. If this keyword is set, the third required argument (`iuvs_output`) becomes unnecessary and is ignored.

- `/new_files`: Query the MAVEN SDC server to determine whether any data files within the window defined by `Time`, exist there and are not currently present locally; then download them if necessary.

- `/inbound`: Return all data within the time range that occurred during the orbit’s inbound leg. This has no bearing on IUVS data, as all observational modes include both inbound and outbound data. It may be used in conjunction with any of the instrument keywords to limit the returned data.

- `/outbound`: Return all data within the time range that occurred during the orbit’s outbound leg. This has no bearing on IUVS data, as all observational modes include both inbound and outbound data. It may be used in conjunction with any of the instrument keywords to limit the returned data.

- **In-Situ instrument flags:**
  The following keywords may be set to return subsets of the data from individual in-situ instruments. Each keyword may be used in conjunction with any of the other in-situ instrument keywords. I.e., they are not exclusive, but additive keywords:
  - `/all_insitu`: return KP data from all in-situ instruments
  - `/lpw`
  - `/euv`
  - `/static`
  - `/swia`
  - `/swea`
  - `/mag`
  - `/sep`
  - `/ngims`

- **IUVS observation mode flags:**
  The following keywords may be set to return only the particular data from each observing mode of the IUVS. Each may be used in conjunction with all other keywords (i.e., they are not exclusive, but additive keywords). IDL is case-insensitive, so the capitalization is useful only for human-reading purposes:
  - `/all_iuvs`: return KP data from all IUVS observation modes
  - `/iuvs_CoronaEchelleHigh`
  - `/iuvs_CoronaEchelleLow`
  - `/iuvs_CoronaEchelleDisk`
  - `/iuvs_CoronaLoresHigh`
  - `/iuvs_CoronaLoresLimb`
  - `/iuvs_CoronaLoresDisk`
  - `/iuvs_StellarOcc`
  - `/iuvs_periapse`
  - `/iuvs_apoapse`
• `/updatePrefs`: Before searching/downloading data, open up a dialogue window to allow the user to update the `mvn_toolkit_prefs.txt` file containing the location of the `ROOT_DATA_DIR`. Once the new path is selected, the search/download will proceed according to the remaining arguments.

• `/only_updatePrefs`: As `/updatePrefs`; but do not attempt to search and download data.

• `/text_files`: Read in ASCII formatted data files (* .tab), rather than the default CDF formatted files (* .cdf).

• `/save_files`: Read in IDL * .sav files (useful for development and debugging), rather than the default CDF formatted files (* .cdf).

• `/debug`: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.

• `/help`: Invoke this list.

### 2.2.2. `mvn_kp_read_model_results`

#### Description
This procedure reads the results of a given simulation result into IDL memory as a structure containing three sub-structures (containing metadata, dimension information, and the model tracers).

#### Example Usage
– Read the Michigan group’s Ionospheric model for Mars Season: 270° and Mean Solar Activity (check these).

```
IDL> mvn_kp_read_model_results, ‘MGITM_LS270_F130.nc’, model
```

– Read the LATMOS group’s Ionospheric model for Mars Season 270°, and Solar Maximum levels of solar activity.

```
IDL> mvn_kp_read_model_results,
   ’Heliosares_Ionoes_LS270_SolMax.nc’, model
```

#### Required Arguments

- **filename**: The file name of the simulation result that is to be read in.

- **output**: This user-defined variable will be the name of the structure returned that contains the simulation results as three sub-structures: meta, containing relevant metadata for the model; dims, containing the values of the dimensions for which the model tracers are provided; and data, containing an array of pointers to structures that contain the name of the tracer, the data parameter, and the order of dimensions in the data (e.g., longitude, latitude, altitude).
List of all accepted Arguments

- **output**: Must be the second argument after the procedure call.
  - Description above in "Required Arguments."
- **/text_files**: Read in ASCII formatted files containing the model outputs. The default is to read in the NetCDF files.
- **/debug**: On error, "stop immediately at the offending statement and print the current program stack." I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
- **/help**: Invoke this list.

2.3. Manipulating Key Parameter Data

Once the MAVEN KP data have been read into IDL memory as a data structure, all data fields may be searched for values that fall within defined parameters. These searches may be run simultaneously. For example, it is possible to use these procedures (see examples below) to find all data records when the spacecraft was between altitudes of 1000km and 2000km, and the STATIC measured O\(^+\) densities greater than 3000 cm\(^{-3}\). Due to fundamental differences between the organization of the in-situ and the IUVS KP data structures, there are separate procedures for searching each data structure.

2.3.1. mvn_kp_insitu_search

**Description**

Search an existing in-situ data structure for data consistent with a set of requirements, and output a new data structure containing only those data consistent with the down-selection.

**Example Usage**

- Find all data records that have a STATIC measured O\(^+\) density greater than 3000 cm\(^{-3}\), and store the results in insitu_out.

  IDL> mvn_kp_insitu_search, insitu_in, insitu_out, parameter='static.oplus_density', min=3000.0

- Find all data records that meet minimum value criteria for two key parameters, identified by index.

  IDL> mvn_kp_insitu_search, insitu_in, insitu_out, parameter=[106,91], min=[3000.0, 150]

- Find all data records that were obtained when the spacecraft was located between 1000km and 3000km altitude.

  IDL> mvn_kp_insitu_search, insitu_in, insitu_out, parameter='spacecraft.altitude', min=1000.0, max=3000.0
**Required Arguments**

**insitu_in:**
The previously created in-situ KP data structure from which a subset of data are to be extracted.

**insitu_out:**
This user-defined variable will be the name of the structure returned that contains all of the requested extracted subset of in-situ KP data.

Either **parameter** or **/list** must be present.

**List of all accepted Arguments**

- **insitu_in:** The input in-situ key parameter data structure produced by a previous call to `mvn_kp_read` or `mvn_kp_insitu_search`.
- **insitu_out:** The output key parameter data structure.
- **/list:** Display an ordered list of all parameters present in the input data structure, `insitu_in`. The items are listed by index, and by instrument followed by name. N.B., if this keyword is present, no down-selection of data based on any provided criteria will be performed, and there will be no output data structure.
- **/range:** List the beginning and end times (and orbits) of the data contained in the passed data structure `insitu_in`. N.B., if this keyword is present, no down-selection of data based on any provided criteria will be performed, and there will be no output data structure.
- **parameter:** Either the name or index (see the **/list** keyword) of the Key Parameter to be searched. This may be a single integer or string, if searching on a single parameter, or an array of integers or strings to search on multiple parameters simultaneously.
- **min:** This is the minimum value for a given search criteria. If not included, then the minimum is assumed to be negative infinity. Like the **parameter** keyword, this may be either a single value (if **parameter** is only a single name/index) or an array of values, where each corresponds to the respective **parameter** name/index.
- **max:** This is the maximum value for a given search criteria. If not included, then the maximum is assumed to be infinity. Like the **parameter** keyword, this may be either a single value (if **parameter** is only a single name/index) or an array of values, where each corresponds to the respective **parameter** name/index.
- **/debug:** On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
- **/help:** Invoke this list.
2.3.2. \texttt{mvn\_kp\_iuvs\_search}

\textbf{Description}

Similar to the previous procedure, this provides the ability for the user to search through an IUVS KP data structure that is stored in memory for records that match a set of criteria. Due to the more complicated nature of the IUVS KP data, this routine is somewhat more complicated than its in-situ counterpart.

\textbf{Example Usage}

- Find all data records that were obtained when the spacecraft was at an altitude greater than 5590km, and store the results in \texttt{iuvs\_out}.

\begin{verbatim}
IDL> \texttt{mvn\_kp\_iuvs\_search, iuvs\_in, iuvs\_out,}
   \texttt{parameter='spacecraft.altitude', min=5590}
\end{verbatim}

- Find all IUVS coronal low resolution limb scans in which the CO$_2$ density was above 1.49x10$^9$cm$^{-3}$.

\begin{verbatim}
IDL> \texttt{mvn\_kp\_iuvs\_search, iuvs\_in, iuvs\_out,}
   \texttt{observation='CoronaLoresLimb', parameter='density',}
   \texttt{species='CO2', min=1490000000}
\end{verbatim}

- Find all IUVS coronal low resolution limb scans in which the CO$_2$ density was above 5.15x10$^9$cm$^{-3}$, and the spacecraft was located between 120km and 220km.

\begin{verbatim}
IDL> \texttt{mvn\_kp\_iuvs\_search, iuvs, iuvs\_out,}
   \texttt{observation='CoronaLoresLimb', parameter='density',}
   \texttt{species='CO2', min=5.15e9, altitude=[120, 220]}
\end{verbatim}

\textbf{Required Arguments}

\begin{description}
\item \texttt{iuvs\_in}: The previously created IUVS KP data structure from which a subset of data is to be extracted.
\item \texttt{iuvs\_out}: This user-defined variable will be the name of the structure returned that contains all of the requested extracted subset of IUVS KP data.
\end{description}

Either \texttt{parameter} or \texttt{/list} must be present.

\textbf{List of all accepted Arguments}

- \texttt{iuvs\_in}: The input IUVS key parameter data structure produced by a previous call to \texttt{mvn\_kp\_read} or \texttt{mvn\_kp\_iuvs\_search}.
- \texttt{iuvs\_out}: The output key parameter data structure.
- \texttt{/list}: Display an ordered list of all parameters present in the input data structure, \texttt{iuvs\_in}. The items are listed by index, and by instrument followed by name. If an observation keyword is not provided, the procedure
will list only “common” variables. I.e., geometry values that exist in all observation modes. If the observation keyword is supplied, then the procedure will also list the tags for that observation mode. N.B., if this keyword is present, no down-selection of data based on any provided criteria will be performed, and there will be no output data structure.

- /range: List the beginning and end times (and orbits) of the data contained in the passed data structure iuvs_in. N.B., if this keyword is present, no down-selection of data based on any provided criteria will be performed, and there will be no output data structure.

- species: Specify a species for which to search. Only applicable if searching for a parameter that has multiple species (e.g., CO₂, CO, H, O, C, N, N₂ for periapse scale height).

- observation: Specify a specific observation as a search criterion. See mvn_kp_read, for a complete list of possibilities.

- parameter: Either the name or index (the full list can be obtained by calling this procedure with the /list keyword, or in Appendix A of this document) of the Key Parameter to be searched. This may be a single integer or string, if searching on a single parameter, or an array of integers or strings to search on multiple parameters simultaneously.

- min: This is the minimum value for a given search criteria. If not included, then the minimum is assumed to be negative infinity. Like the parameter keyword, this may be either a single value (if tag is only a single name/index) or an array of values, where each corresponds to the respective tag name/index.

- max: This is the maximum value for a given search criteria. If not included, then the maximum is assumed to be infinity. Like the parameter keyword, this may be either a single value (if tag is only a single name/index) or an array of values, where each corresponds to the respective tag name/index.

- altitude: A two-element numerical array of altitudes providing the upper and lower bounds on altitude as an additional search criteria. This keyword is only applicable when searching for parameters that are binned by altitude (e.g., periapse radiance).

- /debug: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.

- /help: Invoke this list.

2.3.3. mvn_kp_bin

Description
This routine provides the user with a convenient and efficient way to bin in-situ Key Parameters in one to eight defined dimensions. These guide dimensions may be any of the other Key Parameters within the data structure. The size of each bin is user definable and the output bins may be averages, standard deviations, medians, and quartiles (N.B., quartiles not yet implemented for V20160204).
Example Usage

- Bin the STATIC O\textsuperscript{+} characteristic energy according to spacecraft latitude and longitude, at one degree resolution in latitude, and two degree resolution in longitude. N.B., this assumes the entire KP data table has been read in (data table column format as of 1 June 2015); i.e., indices 177 and 178 may or may not refer to latitude and longitude, use the /list keyword to determine indices associated with each parameter.

\[
\text{IDL}> \text{mvn kp\_bin, insitu\_in, 'static.oplus\_char\_energy',} \\
[177,178], \text{insitu\_out, binsize=[2,1]}
\]

- Bin the SWIA H\textsuperscript{+} density according to spacecraft altitude, with 10km resolution, returning the averaged value (to \texttt{swi\_hplus\_bin\_avg}) and its standard deviation (to \texttt{swi\_hplus\_bin\_std}) in each bin.

\[
\text{IDL}> \text{mvn kp\_bin, insitu\_in, 'swia.hplus\_density',} \\
\text{spacecraft.altitude', insitu\_out, swi\_hplus\_bin\_stddev,} \\
binsize=10, \text{avg\_out=swi\_hplus\_bin\_avg, /std}
\]

Required Arguments

\texttt{insitu\_in}: 
The first argument must be an IDL in-situ key parameter data structure created from \texttt{mvn kp\_read}.

\texttt{to\_bin}: 
The second argument lists the Key Parameter to be binned. Only one key parameter may be binned at a time by this procedure.

\texttt{bin\_by}: 
The third parameter lists the parameters – by index or name – by which to bin the requested key parameter.

\texttt{output}: 
The requested key parameter binned according to the requested dimensions is output to this user-supplied variable. By default, this is the sum of the values of the input data parameter that fall within each bin. To find the average value in each bin, use the \texttt{avg\_out} keyword (see “List of all accepted arguments” below).

\texttt{binsize}: 
Keyword that accepts the array defining the bin size to use for each of the binning dimensions. The number of elements of \texttt{binsize} must equal the number of elements in \texttt{bin\_by}. 
List of all accepted Arguments

- **/list**: Display an ordered list of all parameters present in the input data structure, \texttt{insitu\_in}. The items are listed by index, and by instrument followed by name. N.B., if this keyword is present, no down-selection of data based on any provided criteria will be performed, and there will be no output data structure.

- **list**: When this keyword is provided as a variable, then the assigned variable returns an array of strings containing the list of index and tag names that exist within the input data structure, \texttt{insitu\_in}.

- **/std**: Calculate the standard deviation within each bin and return the information in an array specified by \texttt{std\_out}.

- **std\_out**: Output array containing the standard deviations of the binned key parameter data in each bin. If the \texttt{/std} keyword is not set, nothing is returned to this variable.

- **avg\_out**: Output array containing the average value of the binned key parameter in each bin.

- **mins**: Array of minimum values for each provided binning (number of elements of \texttt{mins} must equal number of elements of \texttt{by\_bin}).

- **maxs**: Array of maximum values for each provided binning (number of elements of \texttt{maxs} must equal number of elements of \texttt{by\_bin}).

- **density**: Array containing the number of values of the input data parameter that fall within each bin.

- **/debug**: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.

- **/help**: Invoke this list.

2.3.4. \texttt{mvn\_kp\_resample}

Description

This routine enables the user to resample a MAVEN in-situ key parameter data structure to an arbitrary time cadence. Used in conjunction with \texttt{mvn\_kp\_add\_data}, this enables the user to modify and extend the KP data (with additional Level-2 data, for example), while retaining the ability to use the plotting and visualization components of the Toolkit.

Example Usage

- Resample the KP data to an hourly cadence.

  \begin{verbatim}
  IDL> hourly = where( (kp\_data.time \texttt{mod} 3600) eq 0 )
  IDL> \texttt{mvn\_kp\_resample, kp\_data, kp\_data[hourly].time, kp\_hourly}
  \end{verbatim}

- Resample the KP data to match the cadence of an NGIMS Level 2 Closed Source Neutral Mode observation (assumed to have been read in to a data structure from the NGIMS Level 2 CSV data file).
IDL> `mvn_kp_resample`, kp_data, ngims_data.t_unix, resampled_KP_data

Required Arguments

**insitu_in:**
The previously created in-situ KP data structure from which a subset of data are to be extracted.

**time:** An array of times to which the input data structure is to be resampled. The accepted units are time in seconds since 1970-01-01T00:00:00 UTC. This routine does not extrapolate, so the time array must be completely within the time range of the input data structure.

**insitu_out:**
This user-defined variable will be the name of the structure returned that contains the input KP data structure, `insitu_in`, resampled to the given time cadence.

List of all accepted Arguments

- **/sc_only:** Only resample the spacecraft data. Mostly, this would be useful for using the visualization routines with non-KP data. By default, all the data within the `insitu_in` data structure will be resampled.

- **/debug:** On error, "stop immediately at the offending statement and print the current program stack." I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.

- **/help:** Invoke this list.

2.4. Plotting Key Parameter Data

These are procedures that will produce “traditional” Abscissa versus Ordinate plots of one or more parameters against time or altitude. In each case, the data structure previously created by `mvn_kp_read` (whether in-situ or IUVS) must be provided to the procedure as an argument.

2.4.1. `mvn_kp_plot`

Description

Plot time series data from a MAVEn in-situ KP data structure. At present, this procedure will not work with IUVS KP data.

Example Usage

- Plot the H+ density from SWIA against time, identifying the parameter by name.

  IDL> `mvn_kp_plot`, insitu, 'swia.hplus_density'
- Create three plots of three attributes, identifying the parameters by KP index number. N.B., the indices will vary according to which instrument data you ingested in `mvn_kp_read`.

```idl
IDL> mvn_kp_plot, insitu, [59, 61, 63]
```

- Create two plots of three attributes, plotting parameter numbers 59 and 61 on one set of axes, and parameter number 63 on another. N.B., IDL will not allow you to create an array with mixed types (i.e., string (‘63’) and integer (63)), so if you get an error while doing this, check your quotes.

```idl
IDL> mvn_kp_plot, insitu, [‘59, 61’, ‘63’]
```

- List all KP data attributes present in a data structure. This is useful if you do not know the name of the attribute you wish to plot, or its index number.

```idl
IDL> mvn_kp_plot, insitu, /list
```

- Create two plots of two parameters with error bars. N.B., measurement errors/uncertainties often are unique KP parameters, generically called “quality flags” because their usage varies across instruments. These are usually immediately following the measured quantity in the table. The code interprets the “quality flags” as defined by the Key Parameters SIS document. *If your needs for displaying uncertainty differ from what is specified there, or if the SIS document and/or the Toolkit are temporarily out of date, it may be necessary to calculate your own error bar quantities and add them to the data structure using `mvn_kp_add_data`.*

```idl
IDL> mvn_kp_plot, insitu, [59, 61], error=[60, 62]
```

- Create a plot of H+ density from SWIA, for a subset of the data contained within the read in data structure, using the `time` keyword to limit the plotted data to orbits between 930 and 932. N.B., an alternate way of doing this would be to use the `mvn_kp_*_search` procedures to create new data structures containing a subset of the originally read data.

```idl
IDL> mvn_kp_plot, insitu, ‘swia.hplus_density’, time=[930, 932]
```

- Generate a plot of in-situ KP parameter 61 with logarithmic axes, either using the `/log` keyword, or passing the IDL plot variable `ylog` to the procedure.

```idl
IDL> mvn_kp_plot, insitu, 61, /log
IDL> mvn_kp_plot, insitu, 61, ylog=1
```

- Generate a plot of spacecraft altitude versus time with a plot window range limited to 1000 to 2000km.
IDL> `mvn_kp_plot`, insitu, `spacecraft.altitude`, yrange=[1000,2000]

**Required Arguments**

**kp_data:**
The first argument must be an in-situ key parameter IDL data structure created from `mvn_kp_read` or `mvn_kp_insitu_search`.

**parameter:**
At least one KP parameter must be provided. This can be either passed as the name(s) of the parameter(s), such as `swia.hplus_density`, or `spacecraft.altitude`, or the index or indices of the parameters, the values of which can be obtained by using the `/list` keyword, described below.

**List of all accepted Arguments**

- **kp_data:** The input in-situ key parameter data structure produced by a previous call to `mvn_kp_read` or `mvn_kp_search`.
- **parameter:** The Key Parameter value(s) to be plotted. The full list can be found in Appendix A, or at the command line by using the `/list` keyword (see below).
- **/list:** Display an ordered list of all parameters present in the data structure. The items are listed by index, and by instrument followed by name. If instead this keyword is assigned to a variable, then the list is stored in that variable as an array of strings.
- **/range:** Display the beginning and ending times and orbits in the data contained in the passed data structure.
- **error:** A scalar or array of indices or strings identifying the parameters to be plotted as error bars in the generated plot(s). The number of parameters supplied to `error` MUST equal the number of parameters supplied to `parameter`. At present, this is a dumb variable: it will make error bars out of whatever you provide to this argument, even if you ask it to plot spacecraft longitude as the error bars to H+ density.
- **time:** Define a range of times to be plotted, this keyword will accept string or integer inputs, as scalars or arrays. See the `time` keyword under “Required Arguments” in `mvn_kp_read` for additional details.
- **/log:** Plot the dependent variable on a logarithmic axis scale.
- **/directgraphic:** Generate plots using traditional (IDL version lt 8.0) direct graphics rather than the default object oriented graphics.
- **plot_object:** to return the plot object to `$MAIN$` to allow further editing of details by the user, assign a variable name to this keyword. This keyword is only applicable if generating object-oriented graphics. The generated figure is stored as an array of plot objects.
- **IDL Plotting keywords:** All regular IDL plotting keywords (except `xrange`, because we have included a separate time parameter to define the time
range plotted) are accepted and passed to the plotting command. N.B., the
definitions of some of these keywords differ whether the plot is generated
via direct graphics (standard for IDL versions up to 7.x) or via object
oriented graphics (standard in IDL version 8.x and greater). Because the IDL
plot procedure/function will be utilizing these keywords (and not the
ToolKit), any errors in parsing the plot keywords will not be handled
gracefully (you will be dumped straight out of the routine in the middle of
the runtime environment). If you would like to back up and start again, enter
the command retall at the command line.

- /debug: On error, “stop immediately at the offending statement and print the
current program stack.” i.e., a less graceful but more informative exit from
the procedure upon the occasion of an error.
- /help: Invoke this list.

2.4.2. mvn_kp_altplot

Description
Generate a plot of in-situ key parameter data plotted versus altitude, rather than
time. Altitude by default is plotted on the y-axis.

Example Usage
- Exactly the same as mvn_kp_plot. Note, however, that some plotting keywords
may behave unexpectedly, since the dependent and independent axes have been
swapped (unless the /davin keyword has been set). For example, to plot three
attributes of the Key Parameter data, identified by index, in three separate sets of
axes:

```
IDL> mvn_kp_altplot, insitu, [59, 61, 63]
```

Required Arguments
See mvn_kp_plot above.

List of all accepted Arguments
- In addition to those listed in mvn_kp_plot, there is also:
- /davin: Reverse the axes. i.e., place altitude on the x-axis and the KP data on
the y-axis

2.4.3. mvn_kp_standards

Description
There are twenty-five standardized plots created from the in-situ KP data found
on the MAVEN SDC website. This procedure generates a direct graphics window
that contains all, or a subset of, those twenty-five plots, from the data provided
by the user. Most of the plotted parameters are directly from the Key Parameter
data; though in some cases, the plotted quantities of interest have been derived.
Example Usage

- Plot all twenty-five standard plots. Normally, you will not want to do this, since this command will generate a single window with 25 rows of very narrow plots. But it might be useful in a quick-look case, so this keyword has been retained.

```idl
IDL> mvn_kp_standards, insitu, /all
```

- Generate a figure containing only three plots: the Magnetic field standard plot in Mars Solar Orbital coordinates (x, y, z, and magnitude) the standard spacecraft ephemeris information (sub-spacecraft latitude/longitude, subsolar latitude/longitude, local solar time, solar zenith angle, and Mars season), and the H*/He++ and pick-up ion omni-directional fluxes from STATIC.

```idl
IDL> mvn_kp_standards, insitu, /mag_mso, /eph_angle, /static_flux
```

- Generate the same figure as the previous example, but change the x-axis to indicate altitude, and customize the title.

```idl
IDL> mvn_kp_standards, insitu, /mag_mso, /eph_angle, /static_flux, /altitude, title='Example 3 of mvn_kp_standards from Users Guide'
```

Required Arguments

**insitu:**

The first argument must be an IDL In-situ key parameter data structure created from `mvn_kp_read`.

Data Selection Keywords:

The following keywords identify which among the data that feed the standardized plots are to be presented in the generated figure. Each of these data selection keywords may be used in conjunction with any of the other data selection keywords. I.e., they are not exclusive, but additive keywords. **At least one of these must be set:**

- `/all`: Generate all 25 plots
- `/euv`: EUV irradiance in each of three bands
- `/mag_mso`: Magnetic field, MSO coordinates
- `/mag_geo`: Magnetic field, Geographic coordinates
- `/mag_cone`: Magnetic clock and cone angles, MSO coordinates
- `/mag_dir`: Magnetic field: radial, horizontal, northward, and eastward components
- `/ngims_neutral`: Neutral atmospheric component densities
- `/ngims_ions`: Ionized atmospheric component densities
- `/eph_angle`: Spacecraft ephemeris information
- `/eph_geo`: Spacecraft position in geographic coordinates
- `/eph_mso`: Spacecraft position in MSO coordinates
- **/swea**: electron parallel/anti-parallel fluxes
- **/sep_ion**: Ion Energy fluxes
- **/sep_electron**: Electron Energy fluxes
- **/wave**: Electric field wave power
- **/plasma_den**: Plasma densities
- **/plasma_temp**: Plasma Temperatures
- **/swia_h_vel**: H⁺ Flow velocity in MSO coordinates from SWIA
- **/static_h_vel**: H⁺ flow velocity in MSO coordinates from STATIC
- **/static_o2_vel**: O₂⁺ flow velocity in MSO coords from STATIC
- **/static_flux**: H⁺/He++ and Pick-up Ion omni-directional fluxes
- **/static_energy**: H⁺/He++ and Pick-up Ion characteristic energies
- **/sun_bar**: Indication of whether MAVEn is in sunlight
- **/solar_wind**: solar wind dynamic pressure
- **/ionosphere**: Electron Spectrum shape parameter
- **/sc_pot**: Spacecraft potential

**List of all accepted Arguments**

- **insitu**: The input in-situ key parameter data structure produced by a previous call to `mvn_kp_read` or `mvn_kp_insitu_search`.
- **time**: This keyword enables the user to plot a subset of the in-situ KP data. By default, all of the data contained within the passed structure are plotted. The user can choose the plotted time range in a number of formats: orbit, date-time string, or double precision UNIX time.
  - **Orbit**: A scalar integer or a two-value integer array that defines either the orbit to be plotted or the range of orbits to be plotted.
  - **Date/time string**: Time in the format yyyy-mm-dd/hh:mm:ss (N.B., PDS standard of yyyy-mm-ddThh:mm:ss is also accepted). If a single date-time string is provided, the procedure will interpret time as a start time. If a two-element array of date-time strings is provided, it will be interpreted as a start-time and an end-time.
  - **UNIX time**: A long integer time interpreted, in the UNIX fashion, as seconds elapsed since 1 January 1970 00:00:00UTC. As with the date-time string format option, if only one time is provided, it is interpreted as a start-time; if two, as a start and end time.
- **/range**: List the beginning and end times (and orbits) of the data contained in the passed data structure `kp_data`. N.B., No data will be plotted if this keyword is provided; all plotting keywords will be ignored.
- **/list**: Display an ordered list of all parameters present in the data structure. The items are listed by index, and by instrument followed by name. If instead this keyword is assigned to a variable, then the list is stored in that variable as an array of strings.
- **/altitude**: Include the spacecraft altitude as a secondary x-axis.
/debug: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.

/help: Invoke this list.

2.4.4. mvn_kp_tplot

Description
Invoke the SSL Berkeley Tplot routines from within the MAVEn IDL Toolkit. A user familiar with Tplot may wish to use this routine to quickly extract MAVEn KP data to Tplot variables.

Example Usage
- List all available Key Parameters within the data structure. N.B, the indexing used by Tplot may differ from that used by mvn_kp_plot and mvn_kp_altplot.

IDL> mvn_kp_tplot, insitu, /list

- Create a single Tplot variable from the Langmuir Probe electron density key parameter data (first line), and then plot it using the SSL Berkeley Tplot software (second line). N.B., invocation of mvn_kp_tplot merely “sets the table” for the Tplot package to then generate the plots.

IDL> mvn_kp_tplot, insitu, parameter='lpw.electron_density'
IDL> tplot,1

- Create all Tplot variables from a given in-situ data structure. Again, note that this command will not generate any plots. Tplot must be later invoked to identify which indices relate to which parameters, and also to plot them. See the Berkeley SSL Tplot documentation for more details on the use of Tplot.

IDL> mvn_kp_tplot, insitu, /createall

- Create Tplot variables for all of the Langmuir probe and SWIA Key Parameter data contained in the passed in-situ data structure.

IDL> mvn_kp_tplot, insitu, /lpw, /swia

Required Arguments
insitu: The first argument must be an IDL In-situ key parameter data structure created from mvn_kp_read.
At least one of /createall or parameter must be provided.

List of all accepted Arguments
- insitu: The input in-situ key parameter data structure produced by a previous call to mvn_kp_read or mvn_kp_insitu_search.
• **time:** This keyword enables the user to plot a subset of the in-situ KP data. By default, the entirety of the data contained within the passed structure is plotted. The user can choose the plotted time range in a number of formats: orbit, date-time string, or double precision UNIX time.
  - **Orbit:** A scalar integer or a two value integer array that defines either the orbit to be plotted or the range of orbits to be plotted.
  - **Date/time string:** Time in the format yyyy-mm-dd/hh:mm:ss, N.B., the PDS standard date/time format, yyyy-mm-ddThh:mm:ss, is also accepted. If a single date/time string is provided, the procedure will interpret time as a start time. If a two-element array of date-time strings is provided, it will be interpreted as a start-time and an end-time.
  - **UNIX time:** A long integer time interpreted, in the UNIX fashion, as seconds elapsed since 1 January 1970 00:00:00UTC. As with the date-time string format option, if only one time is provided, it is interpreted as a start time; if two, a start and end time.

• **/range:** List the beginning and end times (and orbits) of the data contained in the passed data structure `kp_data`. N.B., No data will be plotted if this keyword is provided; all plotting keywords will be ignored.

• **/list:** Display an ordered list of all parameters present in the data structure. The items are listed by index, and by instrument followed by name.

• **Data Selection Keywords:**
  The following keywords identify which among the in-situ instruments contained within the passed data structure are to be accessed for the purpose of creating Tplot variables. Each of these data selection keywords may be used in conjunction with any of the other data selection keywords. I.e., they are not exclusive, but additive keywords:
  - **/createall**
  - **/euv**
  - **/lpw**
  - **/static**
  - **/swea**
  - **/swia**
  - **/mag**
  - **/sep**
  - **/ngims**

• **/prefix:** Override the default action of prepending a prefix of “mvn_kp_” to each Tplot variable.

• **/quiet:** Suppress all text outputs to the screen.

• **message:** Change the Tplot message field printed on each plot generated. By default, this message reads, “Created from MAVEN KP data.”

• **/debug:** On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.

• **/help:** Invoke this list.
2.4.5. mvn_kp_iuvs_limb

Description
Plot the IUVS Periapse limb-scan Key Parameter data. All three limb scans from each orbit are included in each plot, with keywords allowing the user to choose which orbits and species are to be shown. By default, all limb scan data, both radiance and density profiles, are displayed.

Example Usage
- Display all radiance profiles on a single set of axes.
  
  IDL> mvn_kp_iuvs_limb, iuvs, /radiance

- Display all radiance and density profiles simultaneously, on a single set of axes for each.
  
  IDL> mvn_kp_iuvs_limb, iuvs, /radiance, /density

- Display only CO₂ UV doublet, CO Cameron emission, and 1972Å O emission radiance profiles, each on its own set of axes. All profiles within the given IUVS data structure will be plotted.
  
  IDL> mvn_kp_iuvs_limb, iuvs, /radiance, rad_species=[1,2,6], /species_expand

- Display density of all species on individual plots, but only for selected profiles having index numbers 1, 5, 7, and 9. Note that the profiles keyword contains both orbit and profile per orbit information. All profiles selected will be plotted in the same window.
  
  IDL> mvn_kp_iuvs_limb, iuvs, /density, /species_expand, profiles=[1,5,7,9]

- Generate plots of the same data as the previous example; but place each profile and each species on its own set of axes.
  
  IDL> mvn_kp_iuvs_limb, iuvs, /density, /species_expand, profiles=[1,5,7,9], /profile_expand

Required Arguments
kp_data:
The input IUVS key parameter data structure produced by a previous call to mvn_kp_read or mvn_kp_iuvs_search. If the user attempts to pass in-situ data to this routine, an error message will print and the procedure will exit. **At least one of /radiance or /density must be set.**
List of all accepted Arguments

- **/range**: List the beginning and end times (and orbits) of the data contained in the passed data structure kp_data. N.B., No data will be plotted if this keyword is provided; all plotting keywords will be ignored.
- **/density**: Plot the density profiles. Providing both /density and /radiance results in the default behavior (i.e., it plots both).
- **/radiance**: Plot the radiance profiles. Providing both /density and /radiance results in the default behavior (i.e., it plots both).
- **profiles**: if only a subset of profiles contained within the passed data structure is desired, the selected profiles may be identified here in an array. All available profiles will be plotted by default in the absence of this keyword.
- **/profileexpand**: Plot each requested profile on its own set of axes. The default is to plot all profiles on a single set of axes, uniquely identified by color. This keyword applies to both radiance and density profiles. This keyword may be used in conjunction with /speciesexpand to enable various comparisons.
- **den_species**: Plot density profiles for a subset of species by providing an array of allowed IUVS KP limb profile density data species indices. These are:
  1. CO₂/CO₂⁺
  2. CO
  3. H
  4. O
  5. C
  6. N
  7. N₂
- **rad_species**: Plot radiance profiles for a subset of species by providing an array of allowed IUVS KP limb profile radiance data species indices. These are:
  1. CO₂/CO₂⁺ (CO2pUVD)
  2. CO (Cameron)
  3. H
  4. O (1304Å)
  5. O (1356Å)
  6. O (2972Å)
  7. C (1561Å)
  8. C (1657Å)
  9. N (1493Å)
  10. N₂
  11. NO
- **/speciesexpand**: Plot each requested species on its own set of axes. The default is to plot all profiles on a single set of axes, uniquely identified by IDL linestyle. Since there are only six IDL linestyles, the seventh and later plotted species repeat the linestyle sequence but double the line thickness. This keyword applies to both radiance and density profiles. This keyword may be used in conjunction with /profileexpand to enable various comparisons.
- **/info**: List radiance and density species by index number, then return. No plotting is performed if this keyword is present.
- **/nolegend**: Suppress creation of an additional window containing the legend.
- **/linear**: Plot density and/or radiance values on a linear scale. The default is to plot on a logarithmic scale.
- **/log**: Plot density and/or radiance values on a log scale. The default is to plot on a logarithmic scale.
- **/window**: Create the requested plot(s) in a new window. The default is to re-use the currently active window.
- **winx**: Define the window width in pixels. Useful if you are working on a machine with limited display real estate. Currently, this keyword is ignored unless a new window is being created (either with the /window keyword, or the first created plot window on the screen).
- **winy**: Define the window height in pixels. Useful if you are working on a machine with limited display real estate.
- **color_table**: Accepts an integer argument between 0 and 74 and generates the requested plot(s) using the IDL color table identified by the given index. By default, the procedure utilizes color table 40 ('Rainbow + Black'). N.B., the background color for the plot window is defined as the color index 255 in the given color table. This is not always white or black.
- **oo**: to return the plot object to $MAIN$ to allow further editing of details by the user, assign a variable name to this keyword. This keyword is only applicable if generating object-oriented graphics. The generated figure is stored as an array of `<number of profiles> x <number of species>` plot objects.
- **leg**: to return the legend object to $MAIN$ to allow further editing of details by the user, assign a variable name to this keyword. If /nolegend is set, this keyword will be ignored.
- **/debug**: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
- **/help**: Invoke this list.

### 2.4.6. mvn_kp_iuvs_corona

**Description**

Plot the IUVS coronal scan KP data. By default, **all** coronal scan data are displayed.

**Example Usage**

- Plot all IUVS coronal scans contained within the passed data structure.
  
  IDL> `mvn_kp_iuvs_corona, iuvs`

- Plot only the echelle data, and suppress the legend.
  
  IDL> `mvn_kp_iuvs_corona, iuvs, /echelle, /nolegend`

- Plot only the low resolution coronal scans, and suppress printing labels.
IDL> mvn_kp_iuvs_corona, iuvs, /lores, /nolabels

- Plot only the high altitude coronal scans.

IDL> mvn_kp_iuvs_corona, iuvs, /high

**Required Arguments**

**kp_data:**
The input IUVS key parameter data structure produced by a previous call to `mvn_kp_read` or `mvn_kp_iuvs_search`. If the user attempts to pass in-situ data to this routine, an error message will print and the procedure will exit.

**List of all accepted Arguments**

- **/range:** List the beginning and end times (and orbits) of the data contained in the passed data structure `kp_data`. N.B., No data will be plotted if this keyword is provided; all plotting keywords will be ignored.

- **Data Selection Keywords:**
The following keywords identify which among the IUVS coronal scan observation modes contained within the passed data structure are to be accessed for the purpose of generating plots. These modes identify the source of the observation:
  - /echelle: Plot only data obtained using the echelle grating.
  - /lores: Plot only data obtained using the low resolution grating.

While these modes identify the type of observation
  - /disk: Plot only coronal disk data.
  - /limb: Plot only coronal limb data.
  - /high: Plot only coronal high altitude data.

Each of these data selection keywords may be used in conjunction with any of the other data selection keywords. I.e., they are not exclusive, but additive keywords. For now, calling the procedure without any of the data subselection keywords produces plots of all of the available data.

- **/nolabels:** Suppress printing of data labels. Default is to label each plot.

- **/nolegend:** Suppress printing of a plot legend. Default is to generate plot legend(s) in a separate window.

- **colortable:** Accepts an integer argument between 0 and 39 and generates the requested plot(s) using the IDL color table identified by the given index. By default, the procedure utilizes color table 39 ('Rainbow + White'). N.B., the background color for the plot window is defined as the color index 255 in the given color table. This is not always white or black.

- **/window:** Create the requested plot(s) in a new window. The default is to re-use the currently active window.

- **/debug:** On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
2.5. Interpolating Model Results

These are procedures that are designed to read in outputs from simulations of the Martian Ionosphere, Exosphere, and Thermosphere, and interpolate them. The interpolation points can either be a specific altitude, or the spacecraft trajectory of a given in-situ key parameter data structure.

2.5.1. mvn_kp_interpol_model

Description
Given the structure containing the model results read in using `mvn_kp_read_model_results` (or a path to a model file), and an in-situ key parameter data structure as input, this procedure produces a data structure containing all of the simulation’s parameters interpolated to their values at the positions (latitude, longitude, altitude) taken from the spacecraft ephemeris.

Example Usage
- Interpolate all model tracers to the spacecraft trajectory using nearest neighbor interpolation.

```
IDL> mvn_kp_interpol_model, kp_data, model_in, output, /nearest
```

Required Arguments
`kp_data`: The first argument must be an in-situ key parameter IDL data structure created from `mvn_kp_read`, or `mvn_kp_insitu_search`.

`model/file`: The second argument provides the source of the simulation data to be interpolated to the spacecraft trajectory. If model is set, it must be a result from `mvn_kp_read_model_results`. If file is set, then it must be a valid file name and path to a model file.

`output`: The third argument provides the destination of the interpolated model results to the spacecraft ephemeris.

List of all accepted Arguments
- `/debug`: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
- `/nearest`: Uses a nearest neighbor algorithm to determine the values along the spacecraft’s path.
- `/help`: Invoke this list.
2.5.2. mvn_kp_create_model_maps

**Description**
Given the path to a model file, this will generate a png file contour map of a model at a specific altitude. These maps can be used as a background in the 2D or 3D plotting routines.

**Example Usage**
- Interpolate all model tracers to the spacecraft trajectory using nearest neighbor interpolation.

```
>>> mvn_kp_create_model_maps, 170, file = '/path/to/file/MGITM_LS090_F070_150812.nc'
```

**Required Arguments**
- **altitude:**
  The altitude of output map
- **file:**
  If “model” is not provided, you can provide the full path to a model and the script will read in the model file.

**Returns**
A png file will be created in the same directory as the model file provided to the function.

**List of all accepted Arguments**
- **/nearest:** If set, instead of interpolating the nearby values, it will just return the value of the nearest neighbor altitude.
- **/linear:** If set, will perform a simple linear interpolation between two altitude layers. Default is True.
- **transparency:** Numerical value between 0 and 100. Zero is a completely transparent map.
- **ct:** Sets the colortable of the resulting map. For more information, take a look at the IDL ColorTable documentation. The default is COLORTABLE(72, /REVERSE)
- **/fill:** If set, fills in the contour levels rather than generate lines.
- **numContour:** Specify the number of contour lines. The default is 25.

2.6. Adding User-supplied Data

2.6.1. mvn_kp_add_data

**Description**
A simple routine for adding user defined data arrays to the in-situ KP data structure. Arrays may either be added individually, or as part of a structure. If multiple arrays are passed individually, each must be assigned a unique
keyword (see example below), but there is no limit on how many arrays may be added at a time. N.B., The new data arrays added **must** be the same length and cadence as the input structure. If this is not the case, then use `mvn kp resample` to first build a new data structure that matches the length and cadence of the field to be added (or resample or bin the data you wish to add to be consistent with the KP data cadence). All new data fields will be added to a **user** substructure (e.g., `insitu.user.cool_data`). This name is required for later use by the 3D visualization routine.

**Example Usage**

- Add the altitude above the mean Mars radius (rather than the ‘true’ altitude) as a single vector to the in-situ data structure.

  ```idl
  IDL> mso_x = insitu.spacecraft.mso_x
  IDL> mso_y = insitu.spacecraft.mso_y
  IDL> mso_z = insitu.spacecraft.mso_z
  IDL> altitude = sqrt( mso_x^2 + mso_y^2 + mso_z^2 ) - 3390.
  IDL> mvn_kp_add_data, insitu, ‘altitude’, output, data=altitude
  ```

- Suppose you have created simulated data from an external model of the Martian atmosphere that contains O$_2^+$ density, O$_2$ neutral density, CO$_2^+$ density, and CO$_2$ density as four series within a 2D variable called `sim`. Add them to the KP data structure as follows:

  ```idl
  IDL> mvn_kp_add_data, insitu, [‘o2p’, ‘o2’, ‘co2p’, ‘co2’],
  output, data1=sim[0,*], data2=sim[1,*], data3=sim[2,*],
  data4=sim[3,*]
  ```

- Suppose you have created simulated data from an external model of the Martian atmosphere that you have stored in a data structure called `model_out` that you wish to add to the in-situ data structure:

  ```idl
  IDL> mvn_kp_add_data, insitu, tag_names(model_out), output,
  data=model_out
  ```

**Required Arguments**

- **kp_data**: In-situ Key Parameter data structure
- **data_name**: the name(s) given to the new data to be added onto the **user** substructure. The number of elements of **data_name** **MUST** equal the number of variables to be added to the structure.
- **output**: the name of the newly created data structure
- **data**: The data array(s) to be added to the KP data structure.

**List of all accepted Arguments**

- **/debug**: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
2.7. Plotting of Key parameter data in 2D

2.7.1. mvn_kp_map2d

Description
This routine will produce a 2d map of Mars, either in planetocentric or the MSO coordinate system, with the MAVEN orbital projection and a variety of basemaps (including IUVS Apoapse images when available). The spacecraft orbital path may be colored by a given in-situ Key Parameter data value. Additionally, IUVS single point observations may be displayed.

Example Usage
- Plot the spacecraft altitude along the MAVEN orbital track along the surface.

  IDL> mvn_kp_map2d, insitu_data

- Plot the spacecraft altitude along the MAVEN orbital track along the surface, using the MOLA altimetry basemap, and also plot the path of the subsolar point.

  IDL> mvn_kp_map2d, insitu_data, basemap='mola', /subsolarm

- Plot the CO\textsubscript{2}+ density from NGIMS along the MAVEN orbital track along the surface, identifying the parameter by name.

  IDL> mvn_kp_map2d, insitu_data, parameter='ngims.co2plus_density'

- Plot the CO\textsubscript{2}+ density from NGIMS along the MAVEN orbital track along the surface, identifying the parameter by name, using IDL color table #39 (Rainbow+White), and only for densities between 0.1 and 100 cm\textsuperscript{3}

  IDL> mvn_kp_map2d, insitu_data, colors=39, parameter='ngims.co2plus_density', minimum=0.1, maximum=100.

- Plot the CO\textsubscript{2}+ density from NGIMS along the MAVEN orbital track along the surface, limiting the displayed map domain to ±60° latitude and 90° to 270° longitude.

  IDL> mvn_kp_map2d, insitu_data, map_limit=[-60,90,60,270], parameter='ngims.co2plus_density', basemap='mdim'

Required Arguments
- insitu_data: The in-situ Key Parameter data structure
List of all accepted Arguments

- **parameter**: In-situ Key Parameter by which to color the spacecraft trajectory. If not provided, the spacecraft altitude is plotted.
- **minimum**: Minimum value of plotted parameter (or altitude) to display
- **maximum**: Maximum value of plotted parameter (or altitude) to display
- **iuvhs**: The IUVS data structure, needed if the user wishes to plot IUVS data.
- **time**: This keyword enables the user to plot a subset of the in-situ KP data. By default, all of the data contained within the passed structure are plotted. The user can choose the plotted time range in a number of formats: orbit, date-time string, or double precision UNIX time.
  - ** Orbit**: A scalar integer or a two-value integer array that defines either the orbit to be plotted or the range of orbits to be plotted.
  - **Date/time string**: Time in the format yyyy-mm-dd/hh:mm:ss (N.B., PDS standard of yyyy-mm-ddThh:mm:ss is also accepted). If a single date-time string is provided, the procedure will interpret time as a start_time. If a two-element array of date-time strings is provided, it will be interpreted as a start-time and an end-time.
  - **Double precision UNIX time**: A long integer time interpreted, in the UNIX fashion, as seconds elapsed since 1 January 1970 00:00:00UTC. As with the date-time string format option, if only one time is provided, it is interpreted as a start-time; if two, a start and end time.
- **basemap**: The name of the basemap to display upon which the spacecraft data will be overplotted. If not included, a basic lat/lon grid is used as the backdrop. Choices include:
  - 'MDIM': The Mars Digital Image Model.
  - 'MOLA': Mars Topography in color.
  - 'MOLA_BW': Mars topography in black and white.
  - 'MAG': Mars crustal magnetism.
  - 'DUST': IUVS Apopase Dust index image.
  - 'OZONE': IUVS Apopase Ozone index image.
  - 'RAD_H': IUVS Apopase H Radiance image.
  - 'RAD_O': IUVS Apopase O Radiance image.
  - 'RAD_CO': IUVS Apopase CO Radiance image.
  - 'RAD_NO': IUVS Apopase NO Radiance image.
  - 'USER': User-defined basemap. Will open a file dialog window.
- **/nopath**: Suppress the display of the spacecraft orbital track projection.
- **/optimize**: For large data structures, the plotting of the orbital track can get very slow. This keyword decimates the track to a manageable size.
- **/direct**: Forces the use of direct graphics instead of object-oriented.
- **/log**: Colors the spacecraft track with a logarithmic stretch instead of linear.
- **/subsolar**: Plot the path of the subsolar point along the surface of Mars.
- **/msc**: Plot using the MSO map projection. Basemaps are not projected into MSO coordinate systems so only lat/lon grids will be displayed.
• **periapse_temp**: Plot the IUVS periapse temperature measurements on the map along with the spacecraft track.

**Data Selection Keywords:**
The following keywords identify which among the IUVS coronal scan observation modes contained within the passed data structure are to be plotted on the base map. These keywords are recognized **ONLY** if an IUVS Key Parameter data structure has been passed to the procedure, using the `iuvs=` keyword. Each of these data selection keywords may be used in conjunction with any of the other data selection keywords. I.e., they are not exclusive, but additive keywords:

- **/corona_lo_dust**: Plots the Lo-Res coronal dust depth.
- **/corona_lo_ozone**: Plots the Lo-Res coronal ozone depth.
- **/corona_lo_aurora**: Plots the Lo-Res coronal auroral index.
- **/corona_lo_h_rad**: Plots the Lo-Res coronal H radiance.
- **/corona_lo_co_rad**: Plots the Lo-Res coronal CO radiance.
- **/corona_lo_no_rad**: Plots the Lo-Res coronal NO radiance.
- **/corona_lo_o_rad**: Plots the Lo-Res coronal O radiance.
- **/corona_e_h_rad**: Plots the Echelle coronal H Radiance.
- **/corona_e_d_rad**: Plots the Echelle coronal D Radiance.
- **/corona_e_o_rad**: Plots the Echelle coronal O Radiance.

• **/apoapse_blend**: If an IUVS apoapse image is selected as the basemap, this keyword will average all images into a single basemap, instead of plotting only a single image.

• **apoapse_time**: Time of an IUVS Apoapse image to display.

• **color_table**: User-defined color table, either by name (e.g., ‘bw’, or ‘red’), or by IDL color table index (e.g., 1: Blue/White).

• **/list**: Display an ordered list of all parameters present in the data structure. The items are listed by index, and by instrument followed by name. N.B., No data will be plotted if this keyword is provided; all plotting keywords will be ignored.

• **/range**: List the beginning and end times (and orbits) of the data contained in the passed data structure `kp_data`. N.B., No data will be plotted if this keyword is provided; all plotting keywords will be ignored.

• **/debug**: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.

• **/help**: Invoke this list.

2.8. Visualization of 3D data

2.8.1. mvn_kp_3d

**Description**
An interactive 3D visualization of MAVEN spacecraft trajectory and in-situ/IUVS Key Parameter data. This procedure opens a widget, and nearly all of its
capabilities can be accessed via that interface. The options discussed here determine the appearance of the widget and data visualization upon opening. The options that cannot be accessed through the user interface are described in a subsection below.

**Example Usage**

- Fire up the 3D visualization, accessing only in-situ data

  IDL> `mvn_kp_3d, insitu`

- Determine the range of dates within the given in-situ data structure

  IDL> `mvn_kp_3d, insitu, /range`

- Load data between 2015 March 13 and 2015 March 17 from the in-situ data structure, and start the visualization at 2015 March 14 at 9:26:53.

  IDL> `mvn_kp_3d, insitu, time=['2015-03-13T00:00:00', '2015-03-14T09:26:53', '2015-03-17T00:00:00']`

- Display the sub-solar position and the sub-spacecraft position on the globe of Mars at startup. Also, load the MOLA basemap.

  IDL> `mvn_kp_3d, insitu, basemap='mola', /subsolar, /submaven`

- Display the visualization using MSO coordinates, include the Mars-to-Sun vector, and display the orientation of the axes.

  IDL> `mvn_kp_3d, insitu, /sunmodel, /mso, /axes`

- Open the visualization with a plot showing the altitude of the spacecraft with a color bar shown.

  IDL> `mvn_kp_3d, insitu, /parameterplot, /color_bar`

- Open the visualization with a plot showing the CO2+ density data from NGIMS.

  IDL> `mvn_kp_3d, insitu, field='ngims.co2plus_density', /parameterplot, /color_bar`

**Required Arguments**

- **insitu_data**: The in-situ Key Parameter data structure. The in-situ data structure is required because it contains the spacecraft ephemeris information.

**List of all accepted Arguments**

- **iuvs**: The IUVS Key Parameter data structure.
- **time**: If supplied as a scalar or a two-element array, this parameter behaves in the same way as it does in previously discussed plotting routines, such as `mvn_kp_plot`, `mvn_kp_altplot`, etc.
  - If a three element array is provided, then they are sorted, the smallest is assigned as the begin time, the largest as the end time, and the middle is the time displayed as the widget is first created.
  - If the initial display time is not provided, the midpoint time between the begin time and end time is calculated as the default.
  - The order of the provided times does not matter, the earliest will be used as the start time, the latest as the end time, and the middle as the initialization time.
  - Due to the imprecision and degeneracy of parameter values for a given orbit, a three-element input vector of integers representing orbit numbers will not be recognized.

- **parameter**: name of the In-situ Key Parameter to be displayed in the widget at startup. If none is provided, the spacecraft altitude will be displayed.

- **minimum**: the minimum value to be plotted in the orbital trace and the time series plot, if shown. N.B., if the minimum value requested is larger than the minimum value in the data provided, the minimum value of the data will be used instead.

- **maximum**: the minimum value to be plotted in the orbital trace and the time series plot, if shown. N.B., if the maximum value requested is less than the maximum value in the data provided, the maximum value of the data will be used instead.

- **/subsol**: Place a yellow disk at the subsolar point on the surface.

- **/sunmodel**: if selected, will plot the Mars-Sun vector as a yellow line.

- **/submaven**: Place a blue disk at the sub-S/C point on the surface

- **/grid**: Show the lat/lon grid with 30° resolution in latitude and 45° resolution in longitude.

- **/mso**: Plot the spacecraft trajectory in MSO coordinates. Default is in Geometric (longitude, latitude, altitude) coordinates.

- **/axes**: Display the XYZ axes in either MSO or GEO coordinates.

- **/optimize**: For large data structures, the plotting of the orbital track can get very slow. This keyword chooses 5000 data records with equal time index spacing.

- **/showplot**: Upon startup, display a crude trace of the given field (or spacecraft altitude if no field is supplied) as a function of time.

- **/direct**: Generate visualization in direct graphics; but with no interactive capabilities.

- **basemap**: The name of the base map to display upon which the spacecraft data will be overplotted. If not included, a basic lat/lon grid is used as the backdrop. Choices include:
  - 'MDIM': The Mars Digital Image Model.
  - 'MOLA': Mars Topography in color.
- 'MOLA_BW': Mars topography in black and white.
- 'MAG': Mars crustal magnetism.
- 'DUST': IUVS Apoapse Dust index image.
- 'OZONE': IUVS Apoapse Ozone index image.
- 'RAD_H': IUVS Apoapse H Radiance image.
- 'RAD_O': IUVS Apoapse O Radiance image.
- 'RAD_CO': IUVS Apoapse CO Radiance image.
- 'RAD_NO': IUVS Apoapse NO Radiance image.
- 'USER': User defined base map. Will open a file dialog window.

_N.B., for V20160204, due to unavailability of relevant data, the IUVS apoapse maps are not stable._

- **color_table**: Allows the user to provide color table information. If a single integer, the IDL color table with that index is loaded (e.g., 1 for 'Blue/White').
- **bgcolor**: Allows the user to define the background color of “space” in the widget window. If a single byte value (i.e., between 0 and 255) is provided, then a gray between black (0) and white (255) is assigned. If a three-element byte array is provided, it is interpreted as an RGB vector.
- **ambient**: Define the intensity of the flashlight that mimics sunlight. Be aware, a setting of 0.0 does NOT create a crisp terminator.
- **initialview**: Define the initial view of the planet. If a 3-element vector is provided, it is read as [latitude, longitude, radius in km from center]. If a 5-element vector is provided, it is interpreted as [latitude, longitude, radius in km from center, X-coordinate offset in km, Y-coordinate offset in km]. Any other inputs are ignored, after a warning is printed to the screen.
- **scale_factor**: Scale down the size of the widget window for smaller screens.
- **spacecraft_scale**: Change the scale size of the displayed MAVEn spacecraft icon. The default value is 0.03.
- **/list**: Display an ordered list of all parameters present in the data structure. The items are listed by index, and by instrument followed by name. N.B., No data will be plotted if this keyword is provided; all plotting keywords will be ignored.
- **/range**: List the beginning and end times (and orbits) of the data contained in the passed _in-situ_ data structure _kp_data_. N.B., No data will be plotted if this keyword is provided; all plotting keywords will be ignored. If both insitu and iuvs data structures are provided, this returns the range of the in-situ structure (because that is where the relevant ephemeris are stored).
- **/debug**: On error, “stop immediately at the offending statement and print the current program stack.” I.e., a less graceful but more informative exit from the procedure upon the occasion of an error.
- **/help**: Invoke this list.
Arguments not accessible from the widget interface

- `/maven_sphere`: Represent MAVEN with a sphere instead of a model
- `/maven_color`: Change the color of the MAVEN representation
- `/path`: Display the orbit path of MAVEN with a dotted line on the surface
- `/maven_size`: Alter the size of MAVEN

2.9. Checking for Toolkit Updates

2.9.1. `mvn_kp_check_version`

Description

This procedure checks the MAVEN Science Data Center’s website for newer versions of the toolkit. It automatically runs the first time that `mvn_kp_download_files` or `mvn_kp_read` is called in a session, but can also be called manually.

If a newer version is available on the website, the procedure will ask the user if they want to install updates. The procedure will only download and replace the source code files that have changed, so the process completes relatively quickly.

A list of toolkit changes can be found at:


Example Usage

- Check for updates, decline download:

  IDL> `mvn_kp_check_version`
  There is a new version of the software. Would you like to download it now (y/n)? : n

Required Arguments

None.

List of all accepted Arguments

- None.
A. Appendix: KP Data Structures in the IDL ToolKit

The Key Parameter data read in by \texttt{mvn kp read} are held within IDL as a structure of arrays. The name of the structure is defined by the user at the time that \texttt{mvn kp read} is used to read the raw Key Parameter data files, or at the time that \texttt{mvn kp insitu search} or \texttt{mvn kp iuvs search} are used to select a subset of an existing structure of Key Parameter data. For the rest of this section, this name will be assumed to be \texttt{kp data}. Sub-structures are created and defined for each instrument. Not all substructures will be present, according to the subsetting performed either during the \texttt{mvn kp read} or \texttt{mvn kp search}. Full in-situ and IUVS Key Parameter data structures have the following form. Data arrays are in lower case, while structures are listed in CAPS and boldface. For more information, refer to Table 13 of the MAVEn In-Situ Instruments Key Parameters SIS document.

- **KP_DATA**
  - **INSITU**
    - time_string
    - time
    - orbit
    - inbound/outbound flag
  - **SPACECRAFT**
    - GEO_x
    - GEO_y
    - GEO_z
    - MSO_x
    - MSO_y
    - MSO_z
    - GEO_longitude
    - GEO_latitude
    - SZA
    - Local_time
    - Altitude
    - Attitude_geo_x
    - Attitude_geo_y
    - Attitude_geo_z
    - Attitude_mso_x
    - Attitude_mso_y
    - Attitude_mso_z
    - Mars_season
    - Mars_sun_distance
    - Subsolar_point_GEO_longitude
    - Subsolar_point_GEO_latitude
    - SubMars_point_Solar_longitude
    - SubMars_point_solar_latitude
  - **APP**
• Attitude_geo_x
• Attitude_geo_y
• Attitude_geo_z
• Attitude_mso_x
• Attitude_mso_y
• Attitude_mso_z

- **LPW**
  - Electron_density
  - Electron_density_qual_min
  - Electron_density_qual_max
  - Electron_temperature
  - Electron_temperature_qual_min
  - Electron_temperature_qual_max
  - Spacecraft_potential
  - Spacecraft_potential_qual_min
  - Spacecraft_potential_qual_max
  - Ewave_low
  - Ewave_low_qual
    - “0” = perfect
    - “100” = 100% error
  - Ewave_mid
  - Ewave_mid_qual
    - “0” = perfect
    - “100” = 100% error
  - Ewave_high
  - Ewave_high_qual
    - “0” = perfect
    - “100” = 100% error

- **EUV**
  - Quality flags contain: “0” = good data, “1” = off-nominal pointing, “2” = aperture closed
  - irradiance_low
  - irradiance_low_qual
  - irradiance_mid
  - irradiance_mid_qual
  - irradiance_lyman
  - irradiance_lyman_qual

- **MAG**
  - All quality flags are 0=normal data; 1=abnormal data.
  - MSOx
  - MSOx_qual
  - MSOy
  - MSOy_qual
  - MSOz
- MSOz_qual
- GEOx
- GEOx_qual
- GEOy
- GEOy_qual
- GEOz
- GEOz_qual
- RMS
- RMS_qual

**NGIMS**
- All densities are abundances or upper limits in cc\(^{-1}\). All quality flags are % error (1 sigma). Quality flag of “-1” indicates density is an upper limit.

- He_density
- He_density_qual
- 0_density
- 0_density_qual
- CO_density
- CO_density_qual
- N2_density
- N2_density_qual
- NO_density
- NO_density_qual
- Ar_density
- Ar_density_qual
- CO2_density
- CO2_density_qual
- O2plus_density
- O2plus_density_qual
- CO2plus_density
- CO2plus_density_qual
- NOplus_density
- NOplus_density_qual
- Oplus_density
- Oplus_density_qual
- CONplus_density
- CONplus_density_qual
- Cplus_density
- Cplus_density_qual
- OHplus_density
- OHplus_density_qual
- Nplus_density
- Nplus_density_qual

**SEP**
Energy fluxes and their quality flags are in units of eV/cm²/s. Quality flags are standard uncertainty in ion energy flux based on Poisson statistics.

- Ion_energy_flux_1
- Ion_energy_flux_1_qual
- Ion_energy_flux_2
- Ion_energy_flux_2_qual
- Ion_energy_flux_3
- Ion_energy_flux_3_qual
- Ion_energy_flux_4
- Ion_energy_flux_4_qual
- Electron_energy_flux_1
- Electron_energy_flux_1_qual
- Electron_energy_flux_2
- Electron_energy_flux_2_qual
- Electron_energy_flux_3
- Electron_energy_flux_3_qual
- Electron_energy_flux_4
- Electron_energy_flux_4_qual
- Look_direction_1_MSKOx
- Look_direction_1_MSKOy
- Look_direction_1_MSKOz
- Look_direction_2_MSKOx
- Look_direction_2_MSKOy
- Look_direction_2_MSKOz
- Look_direction_3_MSKOx
- Look_direction_3_MSKOy
- Look_direction_3_MSKOz
- Look_direction_4_MSKOx
- Look_direction_4_MSKOy
- Look_direction_4_MSKOz

**STATIC**
- STATIC Quality Flag
- CO2plus_density
- CO2plus_density_qual
- Oplus_density
- Oplus_density_qual
- O2plus_density
- O2plus_density_qual
- CO2plus_temperature
- CO2plus_temperature_qual
- Oplus_temperature
- Oplus_temperature_qual
- O2plus_temperature
- O2plus_temperature_qual
- O2plus_flow_v_appx
- O2plus_flow_v_appx_qual
- O2plus_flow_v_appy
- O2plus_flow_v_appy_qual
- O2plus_flow_v_appz
- O2plus_flow_v_appz_qual
- O2plus_flow_v_MSOx
- O2plus_flow_v_MSOx_qual
- O2plus_flow_v_MSOy
- O2plus_flow_v_MSOy_qual
- O2plus_flow_v_MSOz
- O2plus_flow_v_MSOz_qual
- Hplus_omni_flux
- Hplus_char_energy
- Hplus_char_energy_qual
- Heplus_omni_flux
- Heplus_char_energy
- Heplus_char_energy_qual
- Oplus_omni_flux
- Oplus_char_energy
- Oplus_char_energy_qual
- O2plus_omni_flux
- O2plus_char_energy
- O2plus_char_energy_qual
- Hplus_char_dir_MSOx
- Hplus_char_dir_MSOy
- Hplus_char_dir_MSOz
- Hplus_char_angular_width
- Hplus_char_angular_width_qual
- Dominant_Pickup_ion_char_dir_MSOx
- Dominant_Pickup_ion_char_dir_MSOx_qual
- Dominant_Pickup_ion_char_dir_MSOy
- Dominant_Pickup_ion_char_dir_MSOy_qual
- Dominant_Pickup_ion_char_dir_MSOz
- Dominant_Pickup_ion_char_dir_MSOz_qual
- Dominant_Pickup_ion_char_angular_width
- Dominant_Pickup_ion_char_angular_width_qual

**SWEA**
- Unless noted, quality flags all reflect ‘Statistical Uncertainty’
- Solarwind_e_density
- Solarwind_e_density_qual
- Solarwind_e_temperature
- Solarwind_e_temperature_qual
• Electron_parallel_flux_low
• Electron_parallel_flux_low_qual
• Electron_parallel_flux_mid
• Electron_parallel_flux_mid_qual
• Electron_parallel_flux_high
• Electron_parallel_flux_high_qual
• Electron_antiparallel_flux_low
• Electron_antiparallel_flux_low_qual
• Electron_antiparallel_flux_mid
• Electron_antiparallel_flux_mid_qual
• Electron_antiparallel_flux_high
• Electron_antiparallel_flux_high_qual
• Electron_spectrum_shape
• Electron_spectrum_shape_qual
  - Floating point number from 0 to 1. “Zero” means no evidence for ionospheric electrons, “one” means no evidence for solar wind electrons.

**SWIA**
- Unless noted, quality flags are 0 for bad, and 1 for good, indicating whether the distribution is well-measured and decommutation parameters are definite.

• Hplus_density
• Hplus_density_qual
• Hplus_flow_velocity_MSOx
• Hplus_flow_velocity_MSOx_qual
• Hplus_flow_velocity_MSOy
• Hplus_flow_velocity_MSOy_qual
• Hplus_flow_velocity_MSOz
• Hplus_flow_velocity_MSOz_qual
• Hplus_temperature
• Hplus_temperature_qual
• Solarwind_dynamic_pressure
• Solarwind_dynamic_pressure_qual

**IUVS**

**PERIAPSE**
- Time_start
- Time_stop
- Scale_height
- scale_height_unc
- density
- density_unc
- density_sys_unc
- Radiance
- Radiance_unc
- Radiance_sys_unc
- Temperature
- Temperature_unc
- Sza
- Local_time
- Lat
- Lon
- Lat_mso
- Lon_mso
- Orbit_number
- Mars_season_ls
- Spacecraft_geo
- Spacecraft_mso
- Sun_geo
- Sun_mso
- Spacecraft_geo_longitude
- Spacecraft_geo_latitude
- Spacecraft_mso_longitude
- Spacecraft_mso_latitude
- Subsolar_point_geo_longitude
- Subsolar_point_geo_latitude
- Subsolar_point_mso_longitude
- Subsolar_point_mso_latitude
- Spacecraft_sza
- Spacecraft_local_time
- Spacecraft_altitude
- Mars_sun_distance

- **CORONA_LORES_HIGH**
  - Time_start
  - Time_stop
  - Scale_height
  - scale_height_err
  - density
  - density_err
  - Radiance
  - Radiance_err
  - Temperature
  - Temperature_err
  - Sza
  - Local_time
  - Lat
  - Lon
  - Lat_mso
- Lon_mso
- Orbit_number
- Mars_season_ls
- Spacecraft_geo
- Spacecraft_mso
- Sun_geo
- Sun_mso
- Spacecraft_geo_longitude
- Spacecraft_geo_latitude
- Spacecraft_mso_longitude
- Spacecraft_mso_latitude
- Subsolar_point_geo_longitude
- Subsolar_point_geo_latitude
- Subsolar_point_mso_longitude
- Subsolar_point_mso_latitude
- Spacecraft_sza
- Spacecraft_local_time
- Spacecraft_altitude
- Mars_sun_distance

- **CORONA_LORES_LIMB**
  - Time_start
  - Time_stop
  - Scale_height
  - scale_height_err
  - density
  - density_err
  - Radiance
  - Radiance_err
  - Temperature
  - Temperature_err
  - Sza
  - Local_time
  - Lon
  - Lat_mso
  - Lon_mso
  - Orbit_number
  - Mars_season_ls
  - Spacecraft_geo
  - Spacecraft_mso
  - Sun_geo
  - Sun_mso
  - Spacecraft_geo_longitude
- `Spacecraft_geo_latitude`
- `Spacecraft_mso_longitude`
- `Spacecraft_mso_latitude`
- `Subsolar_point_geo_longitude`
- `Subsolar_point_geo_latitude`
- `Subsolar_point_mso_longitude`
- `Subsolar_point_mso_latitude`
- `Spacecraft_sza`
- `Spacecraft_local_time`
- `Spacecraft_altitude`
- `Mars_sun_distance`

### CORONA_LORES_DISK
- `Time_start`
- `Time_stop`
- `Radiance`
- `Radiance_err`
- `Sza`
- `Local_time`
- `Lat`
- `Lon`
- `Lat_mso`
- `Lon_mso`
- `Orbit_number`
- `Mars_season_ls`
- `Spacecraft_geo`
- `Spacecraft_mso`
- `Sun_geo`
- `Sun_mso`
- `Spacecraft_geo_longitude`
- `Spacecraft_geo_latitude`
- `Spacecraft_mso_longitude`
- `Spacecraft_mso_latitude`
- `Subsolar_point_geo_longitude`
- `Subsolar_point_geo_latitude`
- `Subsolar_point_mso_longitude`
- `Subsolar_point_mso_latitude`
- `Spacecraft_sza`
- `Spacecraft_local_time`
- `Spacecraft_altitude`
- `Mars_sun_distance`

### CORONA_E_HIGH
- `Time_start`
- `Time_stop`
- Lat_mso
- Lon_mso
- Orbit_number
- Mars_season_ls
- Spacecraft_geo
- Spacecraft_mso
- Sun_geo
- Sun_mso
- Spacecraft_geo_longitude
- Spacecraft_geo_latitude
- Spacecraft_mso_longitude
- Spacecraft_mso_latitude
- Subsolar_point_geo_longitude
- Subsolar_point_geo_latitude
- Subsolar_point_mso_longitude
- Subsolar_point_mso_latitude
- Spacecraft_sza
- Spacecraft_local_time
- Spacecraft_altitude
- Mars_sun_distance

**CORONA_E_DISK**
- Time_start
- Time_stop
- Radiance
- Radiance_err
- Sza
- Local_time
- Lat
- Lon
- Lat_mso
- Lon_mso
- Orbit_number
- Mars_season_ls
- Spacecraft_geo
- Spacecraft_mso
- Sun_geo
- Sun_mso
- Spacecraft_geo_longitude
- Spacecraft_geo_latitude
- Spacecraft_mso_longitude
- Spacecraft_mso_latitude
- Subsolar_point_geo_longitude
- Subsolar_point_geo_latitude
- Subsolar_point_mso_longitude
- Subsolar_point_mso_latitude
- Subsolar_point_mso_longitude
- Subsolar_point_mso_latitude
- Spacecraft_sza
- Spacecraft_local_time
- Spacecraft_altitude
- Mars_sun_distance

- **APOAPSE**
  - Time_start
  - Time_stop
  - Ozone_depth
  - Ozone_depth_err
  - Auroral_index
  - Dust_depth
  - Dust_depth_err
  - Radiance
  - Radiance_err
  - Sza_bp
  - Local_time_bp
  - Sza
  - Local_time
  - Lat
  - Lon
  - Lat_mso
  - Lon_mso
  - Orbit_number
  - Mars_season_ls
  - Spacecraft_geo
  - Spacecraft_mso
  - Sun_geo
  - Sun_mso
  - Spacecraft_geo_longitude
  - Spacecraft_geo_latitude
  - Spacecraft_mso_longitude
  - Spacecraft_mso_latitude
  - Subsolar_point_geo_longitude
  - Subsolar_point_geo_latitude
  - Subsolar_point_mso_longitude
  - Subsolar_point_mso_latitude
  - Spacecraft_sza
  - Spacecraft_local_time
  - Spacecraft_altitude
  - Mars_sun_distance