Model Operations Processing System

MOPS v1.01

email: brian@railmops.net

http://www.railmops.net/

Changes

Date	Version	Change
Aug 2010	V1.01	Added two new main menu options – create backup database and purge old print outputs. Added notes on Passenger Car Classes. Added notes on Backup and Print Purge options from main menu (and Housekeeping chapter removed). Note on MOPS Direction added when setting up routes. Adding new parameters DIRECTION and PASSENGER. LSSCHD output updated and notes on TRAIN*, -READY added. Removed Z008 Waybill Completed as this task was now redundant. Information on TRAINS enquiry refreshed.

Introduction

The Model Operations Processing System (MOPS) has been designed to assist working on model railroads and railways. It keeps an organisational view of what's happening on the railroad/railway. It has been designed to have the 'look and feel' of a typical large computer system that would have been in use from the 1960s to the end of the century. It can be used for general railroad and railway operations; although generally the system uses US terminology.

Some experience of computers is required to get MOPS up and running. There is no ‘install’ package that does all the work; rather, there are a series of described steps. This approach has been taken as MOPS can be configured to work in almost any arrangement: the same code runs on most desktop operating systems, and can be used in any combination of Windows, Mac and Linux.

Because of what it's trying to emulate (older mainframe systems), it's a command line entry system – there are no graphics! Although it might take a little time to master the various commands, once they're known it can be faster to use. The system has been designed to be fast and accurate - not pretty.

MOPS is a sophisticated system that supports a range of features that moves modelling of realistic operations a stage closer than simply just running trains:

Industries generate goods for movement;
MOPS replicates loading and unloading times;
Trains can be scheduled, run and monitored;
Cars and Locomotives have maintenance and fuelling requirements;
And MOPS includes its own alert process.

Note that MOPS does NOT link to the railroad itself. It is not intended to actually operate the railroad by computer, but to provide an operational aspect to using the railroad.

Table of Contents

European Union Public Licence V. 1.1

EUPL © the European Community 2007

This European Union Public Licence (the "EUPL") applies to the Work or Software (as defined below) which is provided under the terms of this Licence. Any use of the Work, other than as authorised under this Licence is prohibited (to the extent such use is covered by a right of the copyright holder of the Work). The Original Work is provided under the terms of this Licence when the Licensor (as defined below) has placed the following notice immediately following the copyright notice for the Original Work:

Licensed under the EUPL V.1.1

or has expressed by any other mean his willingness to license under the EUPL.

1. Definitions
In this Licence, the following terms have the following meaning:
- *The Licence*: this Licence.
- *The Original Work* or *the Software*: the software distributed and/or communicated by the Licensor under this Licence, available as Source Code and also as Executable Code as the case may be.
- *Derivative Works*: the works or software that could be created by the Licensee, based upon the Original Work or modifications thereof. This Licence does not define the extent of modification or dependence on the Original Work required in order to classify a work as a Derivative Work; this extent is determined by copyright law applicable in the country mentioned in Article 15.

- *The Work*: the Original Work and/or its Derivative Works.
- *The Source Code*: the human-readable form of the Work which is the most convenient for people to study and modify.
- *The Executable Code:* any code which has generally been compiled and which is meant to be interpreted by a computer as a program.
- *The Licensor*: the natural or legal person that distributes and/or communicates the Work under the Licence.
- *Contributor(s):* any natural or legal person who modifies the Work under the Licence, or otherwise contributes to the creation of a Derivative Work.
- *The Licensee* or *"You":* any natural or legal person who makes any usage of the Software under the terms of the Licence.
- *Distribution* and/or *Communication*: any act of selling, giving, lending, renting, distributing, communicating, transmitting, or otherwise making available, on-line or off-line, copies of the Work or providing access to its essential functionalities at the disposal of any other natural or legal person.

2. Scope of the rights granted by the Licence
The Licensor hereby grants You a world-wide, royalty-free, non-exclusive, sublicensable licence to do the following, for the duration of copyright vested in the Original Work:

- use the Work in any circumstance and for all usage,
- reproduce the Work,
- modify the Original Work, and make Derivative Works based upon the Work,
- communicate to the public, including the right to make available or display the Work or copies thereof to the public and perform publicly, as the case may be, the Work,
- distribute the Work or copies thereof,
- lend and rent the Work or copies thereof,
- sub-license rights in the Work or copies thereof.

Those rights can be exercised on any media, supports and formats, whether now known or later invented, as far as the applicable law permits so.
In the countries where moral rights apply, the Licensor waives his right to exercise his moral right to the extent allowed by law in order to make effective the licence of the economic rights here above listed. The Licensor grants to the Licensee royalty-free, non exclusive usage rights to any patents held by the Licensor, to the extent necessary to make use of the rights granted on the Work under this Licence.

3. Communication of the Source Code
The Licensor may provide the Work either in its Source Code form, or as Executable Code. If the Work is provided as Executable Code, the Licensor provides in addition a machine-readable copy of the Source Code of the Work along with each copy of the Work that the Licensor distributes or indicates, in a notice following the copyright notice attached to the Work, a repository where the Source Code is easily and freely accessible for as long as the Licensor continues to distribute and/or communicate the Work.

4. Limitations on copyright
Nothing in this Licence is intended to deprive the Licensee of the benefits from any exception or limitation to the exclusive rights of the rights owners in the Original Work or Software, of the exhaustion of those rights or of other applicable limitations thereto.

5. Obligations of the Licensee
The grant of the rights mentioned above is subject to some restrictions and obligations imposed on the Licensee. Those obligations are the following:

Attribution right: the Licensee shall keep intact all copyright, patent or trademarks notices and all notices that refer to the Licence and to the disclaimer of warranties. The Licensee must include a copy of such notices and a copy of the Licence with every copy of the Work he/she distributes and/or communicates. The Licensee must cause any Derivative Work to carry prominent notices stating that the Work has been modified and the date of modification.

Copyleft clause: If the Licensee distributes and/or communicates copies of the Original Works or Derivative Works based upon the Original Work, this Distribution and/or Communication will be done under the terms of this Licence or of a later version of this Licence unless the Original Work is expressly distributed only under this version of the Licence. The Licensee (becoming Licensor) cannot offer or impose any additional terms or conditions on the Work or Derivative Work that alter or restrict the terms of the Licence.

Compatibility clause: If the Licensee Distributes and/or Communicates Derivative Works or copies thereof based upon both the Original Work and another work licensed under a Compatible Licence, this Distribution and/or Communication can be done under the terms of this Compatible Licence. For the sake of this clause, "Compatible Licence" refers to the licences listed in the appendix attached to this Licence. Should the Licensee's obligations under the Compatible Licence conflict with his/her obligations under this Licence, the obligations of the Compatible Licence shall prevail.

Background to MOPS

MOPS was largely inspired by TOPS, a mainframe system developed by Southern Pacific in the 1960s. It was subsequently sold to British Rail and other organisations. A fuller history can be obtained from the Internet, at http://en.wikipedia.org/wiki/TOPS.

As described in the introduction, TOPS was a text-based system (having been built in the days before the ubiquitous Graphical User Interface). Text-based systems are used by typing in a command and related data, and the system would process that data accordingly – either updating the system or providing the results of a query. There were no windows, icons, mice or pointers!

There are already a number of systems available, both document-based (eg Car Cards, spreadsheets) processes and computer systems for running Operations on model railroads. The intention of MOPS was to provide another alternative to these systems, and, intentionally, to extend the realism of some aspects of operations by using an 'old-fashioned' system.

An initial paper was produced and published onto the Yahoo Groups Ops site. After obtaining some feedback, a revised paper was produced and further feedback obtained. Much of this feedback has been fed into the resulting system. Further papers and further feedback provided additional input into the proposal.

Python was chosen as the development language. Apart from being multi-platform, its default is a Command Line Interface which meant that the 'feel' of TOPS could be achieved with little additional work. SQLite was chosen as the underlying database: this database has a light footprint and requires little in the way of user maintenance. It has also proven to be efficient in use.

A trial system build was started in Python to evaluate some of the possible technical issues, such as data integrity, cross-platform usage, running of background tasks and so on. Initial results were encouraging, so the build continued in Python. Python is Open Source software and is freely available, as is SQLite. MOPS itself is being released under the EUPL, a public licence. It's therefore free to use, copy and change; there are a number of very small restrictions which are consistent with other free-to-use licences and which simply limit some commercial aspects of its use.

Please, therefore, feel free to improve, change or use MOPS as you require. It has been published under a public licence to allow this to happen – but if you do use it, please let me know either at my email address (on the front cover of this manual) or via the MOPS web site at http://www.railmops.net/

How this Manual works

The Manual is broken down into several parts. The first part deals with setting up MOPS. MOPS uses a scripting language and, because of that, needs software to run it. MOPS also needs a database – SQLite – and, if using Windows, the Python Windows API, which supports printing in Windows. These items are all Open Source software, and this section takes you through obtaining the software and then getting MOPS up and running.

The next section is an overview of the data that MOPS requires. It describes how MOPS works, and allows you to determine what data is required for your particular railroad. Each part of this section segues into dealing with the initial load of data into MOPS. This is the 'static' or reference data – that is, data that pretty well stays the same from session to session.

The third part of the Manual describes 'normal' operational facilities that are available from the application. As well as describing normal operational use, it covers starting a session, correcting data (eg lost cars) and shutting down.

The fourth part of this manual, the Commands Reference section covers all of the commands that can be used on MOPS. As well as describing the command, it provides information on what is needed for each command. A full alphabetical list of commands used by MOPS is also provided.

Getting MOPS Started

NOTE: Because of the wide variation of Operating Systems in use today (all the various flavours of Windows, Macs and Linux), it is not possible to cover each and every installation process. It is assumed that users installing MOPS will have some knowledge of their own machine(s) and are capable of matching the instructions indicated below to their own machine(s).

There may be some small variations between the screens shown in the examples in this guide and those seen in the system. These changes will not be material.

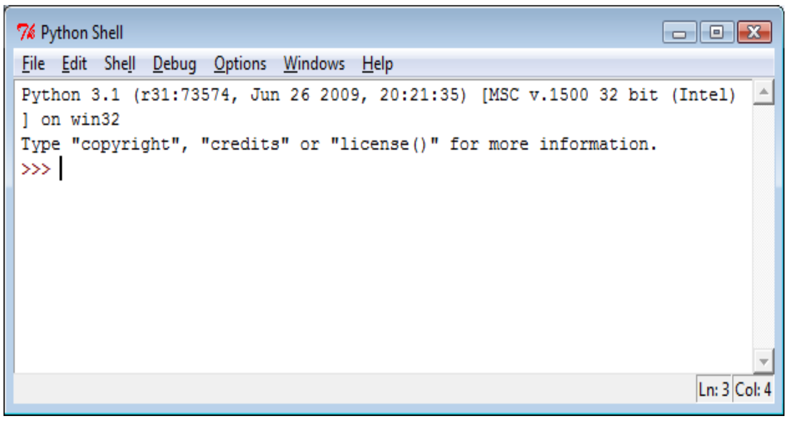
Python

The first step in getting MOPS up and running is to ensure that Python is available on all the machines that are going to use MOPS.

Python can be obtained from http://www.python.org/download/.

The version that should be used is Version 3.1 – download and install this version on each machine (at the time of writing, this version is the most recent release). Versions later than 3.1 will probably work; **earlier versions will not**. Install Python to the default directories.

Check that Python has been correctly installed by selecting IDLE from the Applications menu, which will be in the Python 31 folder. You should see a command window open, identifying the version of Python being run and the Python prompt '>>>':



```
Python 3.1 (r31:73574, Jun 26 2009, 20:21:35) [MSC v.1500 32 bit (Intel)
] on win32
Type "copyright", "credits" or "license()" for more information.
>>> 
```

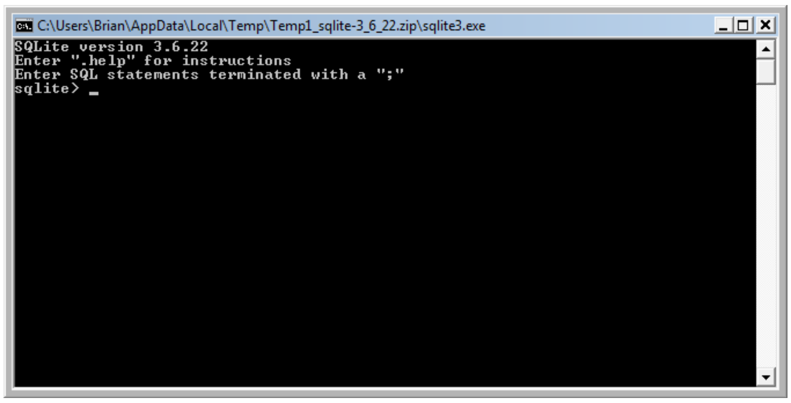
Close the window by typing selecting File then Exit (Windows) or File then Close (Mac) from the menu options. Macs may also ask if you want to kill the process – click on Ok.

Check that you have the current version of Python (some Linux distributions have Python included; ensure that the correct version is being run and update if necessary.

You will also need to download and install SQLite from http://www.sqlite.org/. The version that was used for MOPS is v3.6.22. (For Windows, download the precompiled sqlite binary zipped file for Windows (at the time of writing, this could be found under the ‘Precompiled Binaries For Windows’ on the Download page, and it’s the first zip file shown under Windows (at the time of writing, this was **sqlite-3_6_23_1.zip**).

Linux and Mac users should use their local package managers to download and install the sqlite database software.

Unpack the .zip file and check that the correct version has been obtained by double-clicking on sqlite3.exe. A screen should pop-up (after confirming on a request screen asking whether you should run this software):


C:\Users\Brian\AppData\Local\Temp\Temp1_sqlite-3_6_22.zip\sqlite3.exe

```
SQLite version 3.6.22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> _
```

Type **.quit** to exit the program.

You will also (if using Windows) need to obtain a copy of a specific Python add-on module. This provides additional resources for Windows (XP or Vista) printing and is required for Windows installations if you are going to be printing. The installation can be obtained from //sourceforge.net/projects/pywin32/files/. The version to download is **pywin32-214.win32-py3.1.exe** (for 32-bit Windows): there are other versions available on the download site, but it is necessary to download a version that relates to the Python 3.1 release, which you can identify by checking the 'py3.1' part of the filename. After downloading the file, install Python for Windows.

There is also a fair amount of information available on the Internet about installing and using Python. This can be found at http://www.python.org/doc/current/index.html

Summary Checklist

- Download **Python v3.x**
- Install Python to the default directory
- Check Python runs and a current version is being used
- Download **sqlite**
- Install sqlite to the default directory
- Check sqlite is running the installed module
- Windows: Download and install **pywin32-*** or **pywin64-***

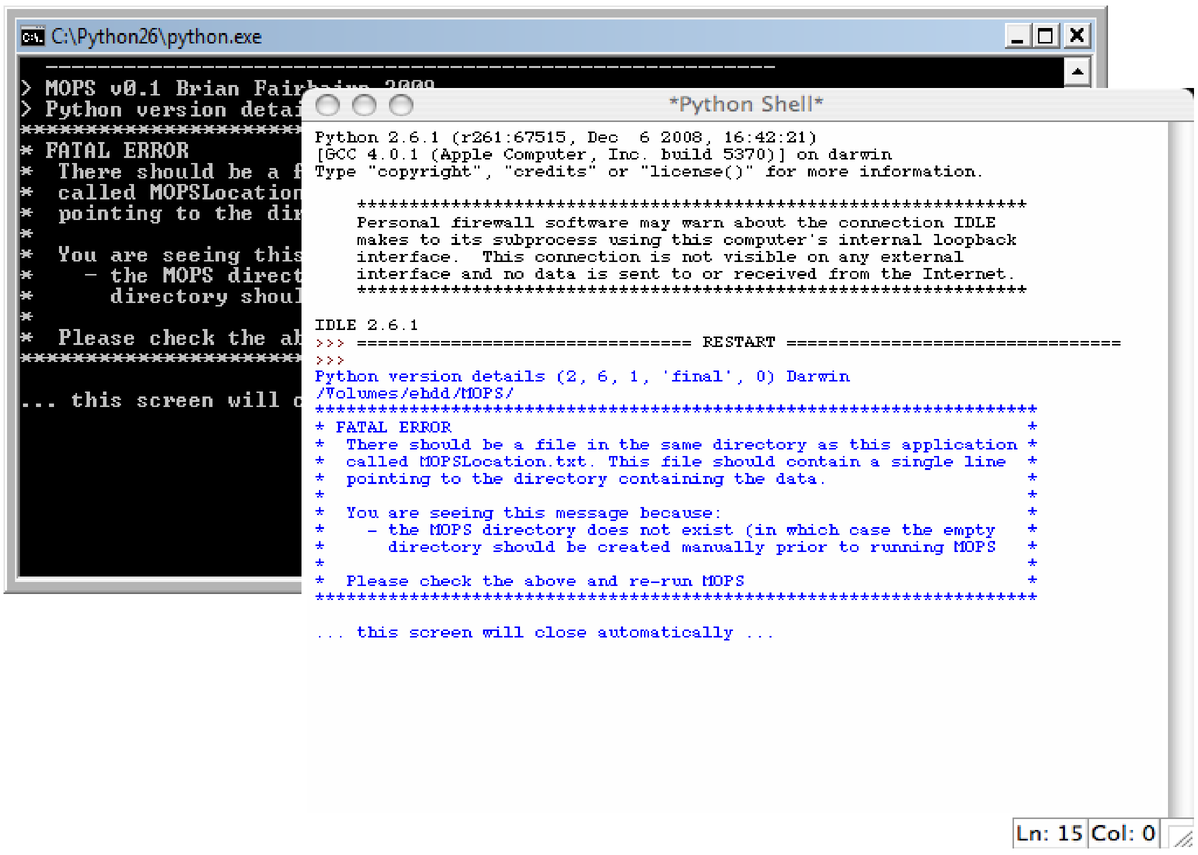
MOPS Application Code

The next step is to download the MOPS Application Code in the .zip file. There are a substantial number of modules that make up the application and these are all included in the zip file.

You should create a directory for the MOPS code – although it can be called anything, it's suggested that it's simply called MOPS. Unpack the contents of the .zip file into that directory – you should see a number of modules ending in .py – these are the Python source code modules. There will be a couple of additional modules as well as the .py modules.

<u>Windows</u>:

Find the file **MOPS.py** and double-click on it. You should information similar to the following screen, which shows that Python has started MOPS correctly:



```
C:\Python26\python.exe
  ------------------------------------------------------------
> MOPS v0.1 Brian Fairb
> Python version detai
*************************
* FATAL ERROR
*  There should be a f
*  called MOPSLocation
*  pointing to the dir
*
*  You are seeing this
*    - the MOPS direct
*      directory shoul
*
*  Please check the ab
*************************

... this screen will c
```

```
*Python Shell*
Python 2.6.1 (r261:67515, Dec  6 2008, 16:42:21)
[GCC 4.0.1 (Apple Computer, Inc. build 5370)] on darwin
Type "copyright", "credits" or "license()" for more information.

    ****************************************************************
    Personal firewall software may warn about the connection IDLE
    makes to its subprocess using this computer's internal loopback
    interface.  This connection is not visible on any external
    interface and no data is sent to or received from the Internet.
    ****************************************************************

IDLE 2.6.1
>>> ================================ RESTART ================================
>>>
Python version details (2, 6, 1, 'final', 0) Darwin
/Volumes/ebdd/MOPS/
**********************************************************************
* FATAL ERROR                                                        *
*  There should be a file in the same directory as this application *
*  called MOPSLocation.txt. This file should contain a single line  *
*  pointing to the directory containing the data.                   *
*                                                                    *
*  You are seeing this message because:                             *
*     - the MOPS directory does not exist (in which case the empty  *
*       directory should be created manually prior to running MOPS *
*                                                                    *
*  Please check the above and re-run MOPS                           *
**********************************************************************

... this screen will close automatically ...

Ln: 15 Col: 0
```

The screen will close automatically after 5 seconds. Don't worry about the Fatal Error at the moment – we are going to fix that in the next section.

<u>**Mac**</u>:
Locate and select the **MOPS.py** file in the directory you have just unpacked. Choose **Get Info** and change the '**Open With**' application to '**IDLE**' (make sure that it's the correct version of IDLE for the version of Python: it may be possible that you can get the Python Launcher to work, in which case that would be the easier route – but this does not, currently, appear to function correctly). Now double-click on **MOPS.py**, a screen will show – and press F5. Again, the screen will close automatically after about 5 seconds. The IDLE application will still be running – this should also be closed.

Linux:
Automatic opening can be configured by editing the MOPS.py source file – there are instructions for configuring the source file available on the web, but as these instructions vary quite significantly for each installation they are not included here.

The slightly longer, but standard, method for starting MOPS is to start the IDLE application, open MOPS.py, then start the module with F5. MOPS will start executing in the Python Shell: you should see the same error information that is displayed for Windows and Mac.

All
The notification about the Fatal Error will be fixed in the next section. For Windows, it's possible t create a short-cut for MOPS.py and, say, move the shortcut to the Desktop. Mac/Linux users can carry out additional work on making the script act as an executable.

Summary Checklist

- Download **MOPS.zip**
- Install MOPS code modules to the required directory
- Carry out the configuration for Windows (shortcut) or Mac (IDLE)
- Mac/Linux: Run MOPS.py using IDLE to check that the code executes
- Windows: Double click on MOPS.py to start the application.
- Carry out any additional, local, modification

MOPS Data

A directory needs to be created to hold the database for MOPS. If the system is being used by a number of users across a network, the directory has to be visible to all users. Users will also need enough access to be able to also update files in that directory. *Make sure that you do not inadvertently create a security issue in your machine or network, especially if you also use your machines or networks for other purposes. Make only those directories that need to be public, public.*

Create or amend, using Notepad or another text editor, the file **MOPSLocation.txt**, which must be located in the directory where the MOPS application code was installed. This file is key to telling MOPS where to find the data. There should only be one entry in the file, and that is the location of the directory where the database is stored. Note that this may take a little figuring out: rather than use MOPS to work out if the directory can be 'seen' or not, use a file explorer program, or a DOS or Terminal window to list files.

If MOPS is being used across more than one computer, then each computer running MOPS needs to have this file **MOPSLocation.txt**, and in each instance will point to the location of the shared data directory as each machine sees it. In this instance, it's likely that the directory location will be different on each computer as it would depend on how directories, etc., are mapped, so this installation is very much up to individual users.

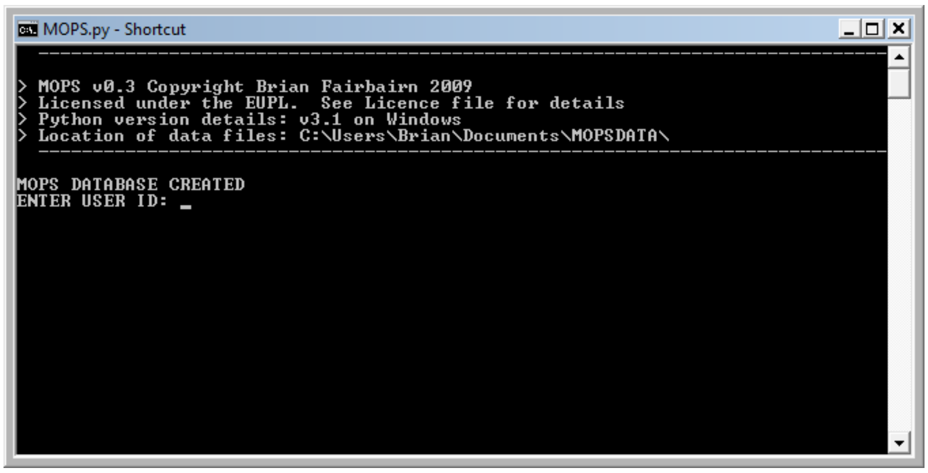
The recommended approach is to get MOPS up and running on the machine on which the database will be stored. Once this is competed and working satisfactorily, additional machines can be added.

A mix of Windows, Mac and Linux computers are supported, so long as each computer has access to the required data directory.

<u>Multiple Railroads:</u> Because the location of the MOPS data directory is held in a file, it is possible to 'swap' data directories being accessed – for example, if a pc is being used on more than one railroad. Simply change the entry in MOPSLocation.txt for the railroad session that you wish to access.

<u>Re-Check MOPS</u>: an initial check can be carried out by re-running MOPS (either through the shortcut on Windows or via IDLE on Mac/Linux, as described above. Starting MOPS with MOPSLocation.txt in place will cause a database (MOPS.db) to be created in the directory indicated in MOPSLocation.txt. The creation of the database will be notified (as well as details about MOPS, Python and the location of the MOPS.db database).

The following screen will be displayed:



```
  ------------------------------------------------------------------------------
> MOPS v0.3 Copyright Brian Fairbairn 2009
> Licensed under the EUPL.  See Licence file for details
> Python version details: v3.1 on Windows
> Location of data files: C:\Users\Brian\Documents\MOPSDATA\
  ------------------------------------------------------------------------------

MOPS DATABASE CREATED
ENTER USER ID: _
```

<u>Summary Checklist</u>

- Create a visible, empty directory for the MOPS data
- **On each machine**, amend MOPSLocation.txt with that directory name
- Check that the Data Directory can be accessed by running MOPS again
- Close MOPS down once more

Loading the Example Data

Along with the MOPS system, an example file has been created that can be loaded up into MOPS. The same data is shown in the examples later in this guide. First, move the file **ds0.txt** (included in the package) into the same directory as indicated by **Location.txt**, and where your data will be stored.

First, start MOPS to get to the User Id prompt: then enter a user if of **xuserx** and a password of **xpassx**:

```
ENTER USER ID: xuserx
ENTER PASSWORD: xpassx
* WARNING - DEFAULT USER ID BEING USED
RAILROAD NAME (RRNAME) CHANGED TO NOT FOUND
>>> MOPS BACKGROUND PROCESSES STOPPED <<<
* SYSTEM STATUS: STOPPED *
****************************************************************************
* MOPS running in Maintenance Mode.  No background tasks are running.      *
****************************************************************************

MOPS NOT STARTED.  PLEASE SELECT:
    1 Start Background tasks for MOPS on this console
    2 Enter MOPS in Maintenance Mode (no background tasks)
    3 Run batch file load (.TXT from data directory
    4 Create Calendar for 5-year period (1950/1955/1960.../2010)
    5 Create back-up of database called MOPS.bak
    6 Purge old MOPS print output files
    9 Quit (this will terminate this session)
==>
```

Select option 4, and enter 1970 for the calendar that we want to create. MOPS will pause for a few seconds, then display a list of messages indicating that it has created a calendar for each year from 1970 to 1974 before taking you back to the menu prompt:

```
==> 4
Enter start year 1950/1955/1960/../2010 ==> 1970
calendar generated for 1970
calendar generated for 1971
calendar generated for 1972
calendar generated for 1973
calendar generated for 1974

MOPS NOT STARTED.  PLEASE SELECT:
    1 Start Background tasks for MOPS on this console
    2 Enter MOPS in Maintenance Mode (no background tasks)
    3 Run batch file load (.TXT from data directory
    4 Create Calendar for 5-year period (1950/1955/1960.../2010)
    5 Create back-up of database called MOPS.bak
    6 Purge old MOPS print output files
    9 Quit (this will terminate this session)
==>
```

This time, select option 3, and enter **ds0** as the prompted filename (without the .txt suffix):

```
==> 3
ENTER FILENAME > ds0
```

And press Enter. If the prompt returns ‘No such file exists...’ then it’s likely that either the **Locations.txt** file is not pointing to the correct location, or the **ds0.txt** file is not in that directory location.

If all is well, then the file will be quickly processed – you may see a lot of error messages but these can be ignored. Once the file has been processed, select **option 9 to log out, then log back in again** and repeat the process. This time, processing the file will be a lot slower (the first time through, the file initiates the code field sizes that are on the file; the second time through, the data itself is loaded).

When the file has loaded, select option 2, enter `liloco` and press Enter: you should see a list of locos that have been loaded into MOPS displayed onto the screen, without any associated error messages (when you see the '+' prompt, just press Enter again):

```
>liloco
12:00 LIST LOCOMOTIVES BY DEFAULT ON 25JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
LOCO TYPE== ENGINE   RR= HOME
1010 GP38/2 DIESEL   APR 1050
1011 GP38/2 DIESEL   APR 1050
1012 GP38/2 DIESEL   APR 4100
1013 GP38/2 DIESEL   APR 4100
1014 GP38/2 DIESEL   PRR 6500
1015 GP38/2 DIESEL   PRR 6500
2101 DD40AX DIESEL   APR 4100
2102 DD40AX DIESEL   APR 4100
2103 DD40AX DIESEL   APR 4100
2104 DD40AX DIESEL   APR 4100
2105 DD40AX DIESEL   APR 4100
2106 DD40AX DIESEL   APR 4100
2107 DD40AX DIESEL   APR 4100
2108 DD40AX DIESEL   APR 4100
2109 DD40AX DIESEL   APR 4100
2110 DD40AX DIESEL   APR 4100
3000 GP7    DIESEL   PRR 6500
3001 GP7    DIESEL   PRR 6500
6011 6MUPC  DIESEL   APR 1050
6012 6MUPC  DIESEL   APR 1050
6013 6MUPC  DIESEL   APR 1050
+
LOCO TYPE== ENGINE   RR= HOME
E401 E42AC  ELECTRIC APR 1050
E402 E42AC  ELECTRIC APR 1050
E403 E42AC  ELECTRIC APR 1050
E404 E42AC  ELECTRIC APR 1050
E405 E42AC  ELECTRIC APR 1050
E406 E42AC  ELECTRIC APR 1050
X100 TAMPER DIESEL   APR 1050
 ** END OF DATA:28  RECORDS DISPLAYED **

>
```

It's suggested that you now create your own user id (see next session).

To remove the loaded test data and create your own data, all that's needed is to rename or delete the **mops.db** file in the data directory.

Creating a MOPS User

Initially, there are no users identified on the system. This is the first item that needs to be created.

A default, limited, user id and password are supplied. The principal use for this default user id is to set up other user ids. The default user id is **xuserx** and the associated password is **xpassx**. This user id is always available and has been designed to set up and/or fix a user id that will let you get going.

Start MOPS as described earlier. MOPS will stop and ask for a User Id:

```
  --------------------------------------------------------------------------
> MOPS v1.0 Copyright Brian Fairbairn 2009/2010
> Licensed under the EUPL.  See Licence file for details
> Python version details: v3.1 on Windows
> Location of data files: C:\Users\Brian\Documents\MOPSDATA\
  --------------------------------------------------------------------------
ENTER USER ID:
```

Input **xuserx** as the User Id and hit Enter.

The system will now prompt for a password. Input **xpassx** and hit Enter.

The screen will then show a few more error messages and stop, waiting for a response to a short menu:

```
ENTER USER ID: xuserx
ENTER PASSWORD: xpassx
* WARNING - DEFAULT USER ID BEING USED
RAILROAD NAME (RRNAME) CHANGED TO NOT FOUND
>>> MOPS BACKGROUND PROCESSES STOPPED <<<
* SYSTEM STATUS: STOPPED *
****************************************************************************
* MOPS running in Maintenance Mode.  No background tasks are running.      *
****************************************************************************

MOPS NOT STARTED.  PLEASE SELECT:
    1 Start Background tasks for MOPS on this console
    2 Enter MOPS in Maintenance Mode (no background tasks)
    3 Run batch file load (.TXT from data directory
    4 Create Calendar for 5-year period (1950/1955/1960.../2010)
    5 Create back-up of database called MOPS.bak
    6 Purge old MOPS print output files
    9 Quit (this will terminate this session)
==>
```

This menu is displayed when the background tasks in MOPS are not up and running. The menu allows you to select what you want MOPS to do. For the moment, select 2 and press Enter.

```
==> 2
***********************************************************************
* MOPS running in Maintenance Mode.  No background tasks are running. *
***********************************************************************
NOT FOUND: WELCOME TO MOPS

SIGN-ON IS COMPLETE: TYPE <QUIT> OR <EXIT> TO END SESSION
```

MOPS is indicating it has signed you on and is waiting for your first command. MOPS is up and running, albeit empty of data.

The first thing that you should do is enter your own details as a supervisor. To create your own details as a Supervisor (there are three levels of access - Supervisor has the widest access) input the following details (substituting your own user id and name - a User Id can be up to eight characters long).

The command for a new user is **aduser**, which is followed by a space, then a userid, then a name, and a level of access (set to 'S' for Supervisor. The example below shows a user id of 'CC' being set up for a Colin Crewe (in red) with an access of supervisor.. Don't worry about upper or lower case – because this is replicating an old mainframe system, everything gets translated to upper case. But note that:

- After the command verb (aduser), there is a space
- After each parameter (id, name, access level), there is a semi-colon. Semi-colons are used to split the different bits of data that the commands need. There will be more o that later.

```
>aduser cc;colin crewe;s
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
 ADD USER BY DEFAULT ON 24Jun2010
* MOPS running in Maintenance Mode.  No background tasks are running.
*
NEW USER DETAILS ADDED
USERID:CC NAME:COLIN CREWE ACCESS:STATUS: SUPERVISOR ACCESS
```

You should see that the new details have been added. There will still be some error and warning messages but ignore these.

Type in **quit.** MOPS will close.

```
>quit
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
 EXIT MOPS BY DEFAULT ON 24Jun2010
MOPS EXIT COMMAND RECEIVED FOR SESSION
MOPS terminated at Thu Jun 24 18:41:08 2010
```

Some Basic MOPS Data

Now start MOPS again. This time, enter the new user id you have just created (cc in this example) followed by the password **password**. When a new user is created, a default password of 'password' is used. You are now prompted to enter a new password. Enter your own new password - which you will use from now on - and you will be signed on. In this example a user has signed on as **cc**, entered an initial password of **password**, then MOPS has prompted a change of password (which has been changed to **tqas**).

```
  ---------------------------------------------------------------------------
> MOPS v1.0 Copyright Brian Fairbairn 2009/2010
> Licensed under the EUPL.  See Licence file for details
> Python version details: v3.1 on Windows
> Location of data files: C:\Users\Brian\Documents\MOPSDATA\
  ---------------------------------------------------------------------------
ENTER USER ID: cc
ENTER PASSWORD: password
ENTER NEW PASSWORD: tqas
PASSWORD HAS BEEN CHANGED
* SYSTEM STATUS: STOPPED *
*****************************************************************************
* MOPS running in Maintenance Mode.  No background tasks are running.      *
*****************************************************************************

MOPS NOT STARTED.  PLEASE SELECT:
    1 Start Background tasks for MOPS on this console
    2 Enter MOPS in Maintenance Mode (no background tasks)
    3 Run batch file load (.TXT from data directory
    4 Create Calendar for 5-year period (1950/1955/1960.../2010)
    5 Create back-up of database called MOPS.bak
    6 Purge old MOPS print output files
    9 Quit (this will terminate this session)
==>
```

Again you will see the menu that's presented when a supervisor logs on and when the background tasks in MOPS are not running. For the moment, enter 2 once more and hit Enter, we want to enter MOPS in Maintenance mode.

The one other item we will change in this section is the name of the railroad or railway that is running MOPS (other railroads will be entered later).

The only bit of data that needs to go with the command is the name of the railroad or railway itself. In the example below, the name of the railway being added is the Atlantic and Pacific Railroad. The one thing you cannot do is use a semi-colon in the name (or in any other data entry, as a semi-colon is the separator between the fields). The command for this is chparm; and the additional bits of data that MOPS needs is the name of the parameter that is changing, and the name of the railroad:

```
>chparm rrname;atlantic and pacific railroad
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
 CHANGE PARAMETER BY CC ON 24Jun2010
* MOPS running in Maintenance Mode.  No background tasks are running.
*
RAILROAD NAME (RRNAME) CHANGED TO ATLANTIC AND PACIFIC RAILROAD
```

You will again see a number of warning messages (these will occur until all the minimum data has been entered), but the last message will indicate that the change has been successful.

If you type in the List Parameters command – **liparm** – you will get a list of all the parameter information on file. At the moment this is limited to the name of the railroad and the status of MOPS:

```
>liparm
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
 LIST PARAMETERS BY CC ON 24Jun2010
* MOPS running in Maintenance Mode.  No background tasks are running. *
PARAM=====   DATA============================================================
RRNAME       ATLANTIC AND PACIFIC RAILROAD
STATUS       STOPPED
 ** END OF DATA:2  RECORDS DISPLAYED **
```

Exit MOPS by entering quit. Start MOPS again, and log back on with your user id and new password. If you are having trouble with a user id or password, then the xuserx user will always be available to enable you to set up a further user.

With MOPS started, select a five-year period that you would like MOPS to cover.   This period is covered in ranges, starting from 1 January 1950; then 1 January 1955 and so on. It is possible to run this twice to have consecutive ranges (for example, to cover 1970 to 1980, run it once for 1970 and again for 1975). Select Option 4, and enter the start year. MOPS will take a few seconds, before indicating that it has generated the required calendars for each year. The example below shows calendars being generated for two consecutive periods:

```
MOPS NOT STARTED.  PLEASE SELECT:
    1 Start Background tasks for MOPS on this console
    2 Enter MOPS in Maintenance Mode (no background tasks)
    3 Run batch file load (.TXT from data directory
    4 Create Calendar for 5-year period (1950/1955/1960.../2010)
    5 Create back-up of database called MOPS.bak
    6 Purge old MOPS print output files
    9 Quit (this will terminate this session)
==> 4
Enter start year 1950/1955/1960/../2010 ==> 1970
calendar generated for 1970
calendar generated for 1971
calendar generated for 1972
calendar generated for 1973
calendar generated for 1974

MOPS NOT STARTED.  PLEASE SELECT:
    1 Start Background tasks for MOPS on this console
    2 Enter MOPS in Maintenance Mode (no background tasks)
    3 Run batch file load (.TXT from data directory
    4 Create Calendar for 5-year period (1950/1955/1960.../2010)
    9 Quit (this will terminate this session)
==> 4
Enter start year 1950/1955/1960/../2010 ==> 1975
calendar generated for 1975
calendar generated for 1976
calendar generated for 1977
calendar generated for 1978
calendar generated for 1979
```

Now select option 2, to enter MOPS in Maintenance mode. We want to set the date and time on MOPS. MOPS has its own internal clock, which can be set to run at a normal or an accelerated speed. The MOPS clock *only* runs when the background tasks are running; so to freeze a MOPS time when an operating session ends, you simply stop the background tasks.

To see the parameters involved in setting up the time, enter **settim**, but instead of a parameter just put a question mark:

```
>settim ?
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
 SET MOPS DATE AND TIME BY CC ON 25Jun2010
* MOPS running in Maintenance Mode.  No background tasks are running. *
SETTIM ddmmmyy hh:mm:ss dow
```

After the current error messages (we will remove these shortly), the required input data is displayed: `SETTIM ddmmmyy hh:mm:ss dow` so that you can see what you need to put in at the command. This help feature is available for every command verb. To set the date/time, enter the date, time and day of week (the day of week is required as a 'safety' feature, to stop the date being changed accidently. It may be that operating sessions want to 'jump' timescales, and the safety feature reduces the chance of a wrong date being entered. You don't have to guess, though; if it is wrong, the system tells you the correct answer, so simply retry:). The date must be within the range of the calendars created earlier:

```
>settim 24jun1972 13:52:00 sat
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
* PARAMETER DATETIME NOT IDENTIFIED FOR UPDATE
 SET MOPS DATE AND TIME BY CC ON 25Jun2010
* MOPS running in Maintenance Mode.  No background tasks are running. *
SYSTEM DATE CHANGED TO: 24JUN1972 13:52:00 SAT
```

You can then re-list the parameter file to see the date/time on the database:

```
>liparm
13:52 LIST PARAMETERS BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
PARAM=====    DATA=====================================================
DATETIME      24JUN1972 13:52:00 SAT
RRNAME        ATLANTIC AND PACIFIC RAILROAD
STATUS        STOPPED
 ** END OF DATA:3  RECORDS DISPLAYED **
```

Finally, set the clock speed for MOPS. This can be varied during sessions, if required, and can be changed later, but for now it will be set to a value of 4:

```
>cspeed 4
13:52 SET MOPS CLOCK SPEED BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
CLOCK SPEED CHANGED TO; 4 TIMES NORMAL
```

Notice that, now that the date/time has been set, the error messages have now disappeared!

Summary Checklist

- Sign on to MOPS with the **xuserx** id
- Create a new Supervisor user with the aduser command (**aduser id;name;'s'**)
- Log off from MOPS, then sign on again with the new user id and password
- Set the railroad/railway name with the chparm command (chparm **rrname;railroad**)
- Create the calendars for the required timescales
- Set the MOPS date and time
- Set MOPS clock speed

What does MOPS need to know?

Geographical Data

Railroads and Areas

MOPS allows a number of Railroads to be set up on a system. Railroads 'own' Areas, Locomotives and Cars. The information required for Railroads is a short code and a name. The short code can be 1 to 10 characters in size; the name can be up to 30 characters long. The permitted size of the short code is determined by the user. Any number of Railroads can be set up on MOPS.

An Area is a group of stations. The number of stations in an area, and how areas are used, is up to the operating railroad. Potentially, an area could be the entire model railroad, with other areas represented by the fiddle or staging yards. A large model railroad may have several areas. An area needs an Area code (again, 1-10 characters in size, with the size being defined by the User) and a name (up to 30 characters). The Area must be allocated to a Railway.

To set the Railroad Code size, enter the following command:

```
>chparm railsize;3
13:52 CHANGE PARAMETER BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.
No background tasks are running. *
RAILROAD CODE SIZE (RAILSIZE) CHANGED
TO 3
```

Set the Area size at the same time:

```
>chparm areasize;3
13:52 CHANGE PARAMETER BY CC ON
24JUN1972
* MOPS running in Maintenance Mode.
No background tasks are running. *
AREA CODE SIZE (AREASIZE) CHANGED TO 3
```

Make sure that there's a semi-colon between the size of the field that's being set, and the value that its being set to.

You now need to quit MOPS, and log back in. This is necessary to re-load the sizes of the fields into memory. Changing field sizes is a one-of task so this won't be a regular occurrence. ***A full list of all parameters are given in the sidebar.***

Many of the codes in MOPS can have their field size specified – for example, the size of the Railroad Code field can be set to 3,4 or 5 characters as required – in fact, any size from 1-10. This allows for more flexible coding structures, and many of the lists and reports automatically adjust their column widths.

However, once a size has been set or changed, it is necessary to log out of MOPS and then log back in again, before new data is added. To save time, you should add all the field sizes at one time.

`CHPARM [field];size`

Field – determines size of the field. Required.

areasize –	size of code field for Areas
cartsize –	size of Car Type code field
classize –	size of Car Class code
commsize –	size of field for Commodities
curosize –	size of Customer Routing
loadsize –	size of Loading Code
locosize –	size of field for Loco numbers
loctsize –	size of Loco Type code field
plaxsize –	size of field for Place Codes
railsize –	size of field for Railroads
routsize –	size of field for Routing code
statsize –	size of Station Type Code
staxsize –	size of field for Station Codes
schdsize –	size of field for Schedule code

Size – size of field.

Value must be between 1 and 10. Required.

When logging back in again, select option 2 as in previous sessions. Then, set up a railroad using the *adrail* command:

```
>adrail apr;atlantic and pacific railroad
13:52 ADD RAILROAD BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW RAILROAD ADDED SUCCESFULLY
APR ATLANTIC AND PACIFIC RAILROAD
```

(If you get an error message, `RAILROAD CODE IS GREATER THAN THE ALLOWED SIZE`, then you need to log off MOPS and log back in again). In this example, the short code has been given as APR, and the name of the railroad set to ATLANTIC AND PACIFIC RAILROAD. The size of the short, code, field, is determined by the railsize value we set earlier (the entered code can be smaller, but not larger than, the code size. Note also that MOPS is now running on its own time system: the date and time on the enquiry is now showing the MOPS date and time. However, while MOPS is stalled (the background processes aren't running), the time won't change.

As well as adding Railroad data, it's possible to change and delete records. A full list of commands is given at the end of this guide.

MOPS requires at least one Area to be added. Each Area MUST be linked to a Railroad, using the appropriate Railroad code. Areas are set up using the *adarea* command:

```
>adarea SPR;springfield;apr
13:52 ADD AREA BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW AREA ADDED SUCCESFULLY
SPR SPRINGFIELD ( APR ATLANTIC AND PACIFIC RAILROAD )
```

In this example we've added an area code SPR with a name of SPRINGFIELD, and attached it to the APR railroad. Again, commands exist to change and delete Areas.

To see what's been added so far, use *lirail* to see a list of Railroads, and *liarea* to see a list of areas:

```
>lirail
13:52 LIST RAILROADS BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
RAI NAME==========================
APR ATLANTIC AND PACIFIC RAILROAD
 ** END OF DATA:1  RECORDS DISPLAYED **

>liarea
13:52 LIST AREAS BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
ARE AREA NAME==========================   RAI RAILROAD NAME=============
SPR SPRINGFIELD                           APR ATLANTIC AND PACIFIC RAILROAD
 ** END OF DATA:1  RECORDS DISPLAYED **
```

Stations and Places

MOPS has the concept of Stations and Places.

Stations are used to control and for routing and timetabling purposes. Generally, rolling stock will be ‘at’ a Station (if not on a train) and can be physically moved around within a Station without further reporting to MOPS.

Places are identifiable locations within a Station. A Place is where actions may take place, such as loading or unloading cars, refuelling or maintenance. A loco, car or wagon will be ‘spotted’ at a Place to start a particular action. Places are ‘action’ points within a Station.

How Stations and Places are used will be up to individual railroads. A Place code is only unique within a Station, so Place codes can be re-used across Stations.

For Places, it’s possible to replicate a Zone-Track-Spot structure:

Station	Place (Zone-Track-Spot)
1000	12-00-00 (this is a zone-level entry)
1000	12-20-00 (this is a track-level entry)
1000	12-20-30 (this is a spot-level entry)
1000	12-20-31 (this is a spot-level entry)

Alternatively, a simpler coding structure could be used, where a Place code means something:

Station	Place
1000	01 Industry
1000	02 Another Industry
1000	89 Diesel Refuelling Point
1010	89 Diesel refuelling Point (at a different station)

Or simply create a different station for everything, and only allow one Place for each Station:

Station	Place
1000	1
1010	1
1020	1
1030	1

Because Stations are used for routing purposes, they are also identified with a Station Type. Typically, a Station Type may be a Station, a Junction, Interchange or a Staging or Fiddle yard. The Station Types are simply used to make identification of the nature of the Station more visible, and can be set up as required.

Places also have a type, but these are predefined as particular actions are supported at Places. The different place types are:

- Industry (these places have specialised processing)
- Diesel Refuelling
- Steam Refuelling (MOPS doesn’t distinguish between coal/water)
- Other Refuelling (MOPS doesn’t define what that might be)
- Maintenance of cars
- Cleaning of cars
- Maintenance of locomotives

First, set up the sizes for each of the fields that are going to be added up: Station Types, Stations and Places. These are added up using the *chparm* command as follows:

Station Type:

```
        >chparm statsize;3
        13:52 CHANGE PARAMETER BY CC ON 24JUN1972
        * MOPS running in Maintenance Mode.  No background tasks are running. *
        STATION TYPE CODE SIZE (STATSIZE) CHANGED TO 3
```

Station:

```
        >chparm staxsize;4
        13:52 CHANGE PARAMETER BY CC ON 24JUN1972
        * MOPS running in Maintenance Mode.  No background tasks are running. *
        STATION CODE SIZE (STAXSIZE) CHANGED TO 4
```

Place:

```
        >chparm plaxsize;2
        13:52 CHANGE PARAMETER BY CC ON 24JUN1972
        * MOPS running in Maintenance Mode.  No background tasks are running. *
        PLACE CODE SIZE (PLAXSIZE) CHANGED TO 2
```

And because this is a parameter that's being changed, you need to log out, then log back in again using Option 2. Check the data has all been created by using the *liparm* command:

```
        >liparm
        13:52 LIST PARAMETERS BY CC ON 24JUN1972
        * MOPS running in Maintenance Mode.  No background tasks are running. *
        PARAM=====    DATA=====================================================
        AREASIZE      3
        CSPEED        4
        DATETIME      24JUN1972 13:52:00 SAT
        PLAXSIZE      2
        RAILSIZE      3
        RRNAME        ATLANTIC AND PACIFIC RAILROAD
        STATSIZE      3
        STATUS        STOPPED
        STAXSIZE      4
         ** END OF DATA:9  RECORDS DISPLAYED **
```

MOPS requires at least one Station Type to be added: the *adstat* command is used for this. The Station Type we are going to create is STN – STATION

```
        >adstat stn;station
        13:52 ADD STATION TYPE BY CC ON 24JUN1972
        * MOPS running in Maintenance Mode.  No background tasks are running. *
        NEW STATION TYPE ADDED SUCCESFULLY
        STN STATION
```

Add a Station. In this instance, the Station that's going to be created is going to be given a numeric code of 4100, with a short name of SPNGFDYD and a long name, belong to Area SPR and be a Station Type of STN. **NOTE** that there is a trailing semi-colon – because we are missing an optional field out. We will come to that in a moment. First, a new station is added using the *adstax* command:

```
        >adstax 1050;spngfdyd;springfield yard;spr;stn;
        13:52 ADD STATION BY CC ON 24JUN1972
        * MOPS running in Maintenance Mode.  No background tasks are running. *
        NEW STATION ADDED SUCCESFULLY
        1050   SPNGFDYD SPRINGFIELD YARD
        AREA: SPR SPRINGFIELD  TYPE: STN STATION
```

Finally, add a Place using the *adplax* command. We are going to give Springfield a whole yard: we cold, if we had wanted to, broken down the yard into sections and have arrivals, departures, holding and so on. But in this instance – to keep it simple, we will just add the whole yard. The ‘X’ signifies that there are no specific actions here (it’s not an industry, and it’s not refuelling or maintaining rolling stock).

```
>adplax 1050;88;springfield yard;x;15000
13:52 ADD PLACE BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW PLACE ADDED SUCCESFULLY
STATION:1050 SPRINGFIELD YARD  PLACE:88 SPRINGFIELD YARD (GEN. ID IS 1)
TRACK LENGTH:15000
```

In this instance, we’ve added a place with a code of 88 to station 1050. The place also has a name – SPRINGFIELD YARD and when it’s generated the record it’s given a unique place id of 1.

Additional Geographical Information

Special Station Processing. There's one thing that can be done with Stations that can't be done with anything else: we can give them an alias. This allows a staging or fiddle yard to represent 'several' places. If we add another Station Type – STA for STAGING, and then add another Station 1000 SPRINGFIELD against this Station Type:

```
>adstat sta;staging
13:52 ADD STATION TYPE BY CC ON 24JUN1972
NEW STATION TYPE ADDED SUCCESFULLY
STA STAGING

>adstax 1000;sprngfld;springfield;spr;sta;
13:52 ADD STATION BY CC ON 24JUN1972
NEW STATION ADDED SUCCESFULLY
1000   SPRNGFLD SPRINGFIELD
AREA: SPR SPRINGFIELD  TYPE: STA STAGING
```

Then add another Station, but this time we give that Station as an alias of the first station. Let's add HARTFORD with a code of 1100 and let this be an alias of SPRINGFIELD. This means that MOPS can 'send' cars to HARTFORD but 'knows' that they are at SPRINGFIELD:

```
>adstax 1100;hartford;hartford;spr;sta;1000
13:52 ADD STATION BY CC ON 24JUN1972
NEW STATION ADDED SUCCESFULLY
1100   HARTFORD HARTFORD
AREA: SPR SPRINGFIELD  TYPE: STA STAGING
ALIAS OF: 1000 SPRINGFIELD
```

Place Types. In addition to the unclassified place type X, the following types of places are available on MOPS. These places perform actions when the appropriate cars or locomotives are placed at these locations. Although Industries are also Places, these are dealt with later. Other Rolling Stock can be placed at these locations; it's just that nothing will happen there.

Type Code	Actions	Affects
C	Clean Cars	All Cars
D	Refuel Diesel Locomotives	Diesel Locomotives only
L	Maintain Locomotives	All Locomotives
M	Maintain Cars	All Cars
O	Other Refuelling	'Other' Locomotives only
S	Steam Locomotive Locomotives	Steam Locomotives only
X	No Facilities	All Rolling Stock

Place Lengths. One of the facilities in MOPS is to track lengths of tracks and trains, which means that the lengths of cars and locomotives are also stored. However, to be of any use, the length measurement must be consistent whenever a length is used: this could be feet, yards, meters or even a specific or tailored length – or even setting cars and locos to be '1' unit long and expressing track lengths simply as a number of cars or locos allowed. The only requirement is that it must be an integer: no fractional numbers are allowed.

Geographical view. One additional enquiry, which lets you see where everything is in relation to each other, is the *ligeog* enquiry. This shows areas within railroads; stations within areas, and places within stations:

```
>ligeog
13:52 PRINT GEOGRAPHY BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
APR ATLANTIC AND PACIFIC RAILROAD
    SCH SCHODACK
        7000 SCHODACK
        7100 ALBANY
        7200 SYRACUSE
        7300 ROCHESTER
    SPR SPRINGFIELD
        1000 SPRINGFIELD
        1050 SPRINGFIELD YARD
            75 SPRINGFIELD YARD ARRIVALS (+10)
            76 SPRINGFIELD YARD HOLDING (+11)
            77 SPRINGFIELD YARD DEPARTS (+12)
            78 SPRINGFIELD YARD RIP TRCK (+13)
        1100 HARTFORD
            85 RUSSEL ROAD DIESEL DEPOT (+3)
        1200 BOSTON
        1300 PROVIDENCE
        2100 RUSSEL ROAD
            81 RUSSEL ROAD RIP TRACK (+2)
            87 RUSSEL ROAD RIP TRACK (+1)
```

Industrial Data

Commodities, Loading and Unloading

MOPS doesn't allow Commodities simply to be loaded onto any convenient car or wagon. In addition to just having a car at the right place ready to be loaded, the loading capability of the car must be the same as the loading method of the commodity. And it must be possible to use that loading method at that place; similarly, a car can be unloaded at a place if the wagon and the place agree on an unloading code.

Because Loading Codes are maintained by the user, it's possible to set a range of validation from allowing everything to be unloaded everywhere (set one Loading and Unloading code for 'everything') to being very specific (eg three-hopper gravity chutes).

Loading Validation: the commodity must be capable of being loaded into a car at that place
    Commodity, Industry and Car all must have the same Loading code

Unloading Validation: the car must be capable of being unloaded at that place:
    Industry and Car must have the same Unloading code

Commodities also have loading and unloading rates. It's necessary at this point to work out what units are going to be used for weight (in the same way that units had to be decided for length, earlier). Weight is used for:

- Commodities - loading and unloading rates (weight/hour)
- Weight of commodity loaded into a car
- Weights of cars (unladen weight)
- Weights of trains (cars plus unpowered locomotives)
- Haulage power of Locos (strictly speaking, not a weight)

Recommended units would be thousands of pounds (MOPS restricts the size of numeric fields, primarily to reduce widths when displaying information as MOPS limits itself to an 80-character display), tons or tonnes, or a home-grown unit. Whichever unit is chosen, it must be capable of being dealt with throughout the system in units or multiples of units eg a commodity can't be loaded at 0.4 units/hour.

The Loading and Commodity code field sizes need to be prepared – the size of the fields for the commodity and loading codes is determined by the user:

```
>chparm loadsize;4
13:52 CHANGE PARAMETER BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
LOADING CODE SIZE (LOADSIZE) CHANGED TO 4

>chparm commsize;3
13:52 CHANGE PARAMETER BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
COMMODITY CODE SIZE (COMMSIZE) CHANGED TO 3
```

When these have been added, log off MOPS and log back on again.

Add a Loading Code. A Loading code can be used for both loading and unloading; or, alternatively, for either loading or unloading only. For example, a ramp or platform can be used for loading and unloading, whereas a car unloading through bottom dump doors couldn't obviously be loaded the same way.

```
>adload ramp;loading platform;y;y
13:52 ADD LOADING DEFINITIONS BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW LOADING CODE ADDED SUCCESFULLY
RAMP: LOADING PLATFORM
LOADING:YES UNLOADING:YES
```

In this instance a LOADING PLATFORM has been added with a code of RAMP. It's capable of being used for both Loading and Unloading.

A Commodity requires a little more information. As well as having a code and a description, it is also associated with a loading code (but not an unloading code, once the commodity is in the car, we don't need to know whether it will get out – the assumption is that, somehow, it will).

We also need to know how fast (weight/hour) it will be to load and unload the car. This isn't necessarily just a feature of the commodity/car: it's how long we expect the customer to take in unloading the car. The car might take an hour to unload once the fork lift truck arrives – but the loading and unloading rates reflect *how long the car will be with the customer* and we want to reflect loading rates as 'time with customer' rather than 'physical time to load'.

Finally, we need to know if the commodity requires clean cars. Cars have a 'Clean Car' indicator indicating whether they have been cleaned or not.

```
>adcomm 100;bagged grain on pallets;ramp;5;4;y
13:52 ADD COMMODITY BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW COMMODITY ADDED SUCCESFULLY
100: BAGGED GRAIN ON PALLETS
LOADING CODE:RAMP LOADING PLATFORM
LOADING RATE:5 UNLOADING RATE:4
COMMODITY REQUIRES CLEAN CARS
```

RAMP is, of course, the previously entered loading code. The numbers 5 and 4 are the loading and unloading codes, respectively, and the Y at the end indicates that clean cars are required.

Industries and Warehouses

Industries are located at Places (there can only be one industry at each place). Each Industry has one or more warehouses attached to it – this allows industries to produce more than one type of goods.

Crucially, the Loading and Unloading codes are associated with the Industry, not the Warehouse. So it's not possible to have a Ramp and a Chute at the same Industry; these would have to be identified as two separate industries. However, it would be possible to have an Industry that produced drums of tomato sauce and pallets of tomato ketchup that were both loaded by ramp, by setting a Warehouse for each commodity.

If there is a customer that has a large number of facilities, then that customer will have to be represented by a number of Industries, with the Industry Name coded up appropriately – for example, a customer FRANKS INC might be coded up as FRANKS#1, FRANKS#2, etc.

Warehouses are linked to an Industry and are associated with the production of the commodity. They determine what commodity is produced, and how quickly. When the amount of commodity exceeds a threshold level, Warehouses automatically call for one or more empty cars of a particular class.

They also hold information about the destination of the commodity (which is another Industry). A Warehouse can only be associated with one commodity and one destination: if multiple destinations and/or multiple commodities are required then additional warehouses have to be created for each required combination.

Finally, each Warehouse can be associated with a Customer Routing code. This is the route that the car is expected to take to its destination. This routing is NOT associated with set-up of routes and schedules later in this guide: this routing is what would be displayed on the customer's instructions or waybill.

Set up the size of the code fields for the customer routing. Industries codes have the same size of field as Places, and Warehouse codes are generated automatically by the system:

```
>chparm curosize;3
13:52 CHANGE PARAMETER BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
CUSTOMER ROUTING CODE SIZE (CUROSIZE) CHANGED TO 3
```

At this stage, we also need to identify what car classes will be used to service the Industry. We will be coming back to Car Classes later, but in the meantime set up the Car Class code field size:

```
>chparm classize;3
13:52 CHANGE PARAMETER BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
CAR CLASS CODE SIZE (CLASSIZE) CHANGED TO 3
```

And log off MOPS and log back on again to pick up the changed parameter.

The next step is to add an Industry. First, add a new Station for Russell Road:

```
>adstax 2100;russelrd;russell road;spr;stn;
13:52 ADD STATION BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW STATION ADDED SUCCESFULLY
2100   RUSSELRD RUSSELL ROAD
AREA: SPR SPRINGFIELD  TYPE: STN STATION
```

Then add an industry at Russell Road:

```
>adindy 2100;01;mountain high aluminium;450;mountainal;ramp;
13:52 ADD INDUSTRY BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW INDUSTRY ADDED SUCCESFULLY: MOUNTAILAL
STATION:2100 RUSSEL ROAD  PLACE:01 MOUNTAIN HIGH ALUMINIUM (GEN. ID IS 22)
TRACK LENGTH:450
  LOAD:RAMP (LOADING PLATFORM)
```

We need to add a second Industry to be the destination of the commodity, so a new Station 5000 FELHAM JUNCTION is added, along with another industry:

```
>adindy 5000;01;felham canning;130;felhamcan;ramp;ramp
13:52 ADD INDUSTRY BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW INDUSTRY ADDED SUCCESFULLY: FELHAMCAN
STATION:5000 FELHAM JUNCTION  PLACE:01 FELHAM CANNING (GEN. ID IS 23)
TRACK LENGTH:130
  LOAD:RAMP (LOADING PLATFORM)
UNLOAD:RAMP (LOADING PLATFORM)
```

And we need to add a Car Class, indicating the type of cars that the Warehouse will order:

```
>adclas box;box car
13:52 ADD CAR CLASSIFICATION BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW CAR CLASS ADDED SUCCESFULLY
BOX BOX CAR
```

Before we add a Warehouse, we need to set up a routing code – this is simply a code with some text after it: the text can actually be anything you want it to be:

```
>adcuro 002;ptsfld-ptfdic-stckbr-sprfld-prvdnc
13:52 ADD CUSTOMER ROUTING INFORMATION BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW CUSTOMER ROUTING ADDED SUCCESFULLY
002 PTSFLD-PTFDIC-STCKBR-SPRFLD-PRVDNC
```

So we now have an industry MOUNTAIN HIGH ALUMINIUM with a track length of 450 feet which has a LOADING PLATFORM. We've given the Industry a name of MOUNTAINAL – we can have up to 10 characters for an industry name. The Industry will produce commoditise which will be sent to FELHAM CANNING (Industry name FELHAMCAN) in BOX CARS. The route that the cars should take is PTSFLD-PTFDIC-STCKBR-SPRFLD-PRVDNC

And now we can create a Warehouse:

```
>adware mountainal;600;felhamcan;4;220;3;box;2200;002
13:52 ADD WAREHOUSE BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW WAREHOUSE ADDED SUCCESFULLY
ID:1 MOUNTAINAL MOUNTAIN HIGH ALUMINIUM
COMMODITY:600 ALUMINIUM SHEETS
DESTINATION:FELHAMCAN FELHAM CANNING UNLOADING: RAMP
PRODUCTION:4 MAX STORAGE:2200 ORDER AT:220 FOR:3 CAR(S) OF TYPE:BOX
ROUTING:002 (PTSFLD-PTFDIC-STCKBR-SPRFLD-PRVDNC)
```

The Warehouse will generate 4 tons per hour of Aluminium sheets at MONTAINAL for sending to FELHAMCAN. They will be loaded into BOX CARS. When the amount of material on hand exceeds 220 tons, an order for 3 BOX cars will be requested. The Warehouse will cease production if there are more than 2200 tons on hand.

Warehouses generate goods *each hour* – so when working out what the rate of production is, this should be borne in mind. Cars loading rates are *per minute*. The discrepancy between the time rates used is to allow for loading of coal hoppers (which can load tons per minute).

Warehouse production can be quickly changed with a single command *cpware*, so if there are difficulties in balancing production and cars this is the first port of call. Additionally, the warehouses understand a threshold value, after which they stop work!

One of the more difficult issues to get right on operating a model railroad is the balance between production, car demands and car availability.

If an existing operating process is not being used – CarCards, another computer system – then the suggestion is to create only one warehouse to begin with and to slowly work up the number of warehouses. At the start, this will mean a lot of cars stuck in yards, but building the traffic flows one warehouse at a time should highlight any problems.

If an existing system is being used, then the suggestion is to replace one warehouse at a time, running both systems alongside each other. Remember that in the real world, it's not uncommon for the introduction of new computer systems to cause additional work over a conversion period!

Rolling Stock

Locomotive Types and Locomotives

Locomotive Types contain the definitions of classes of locomotives. Details such as weight, haulage power, length and so are all attributes of a locomotive class. Individual locomotives themselves will contain, in addition to their running or road number, information about their current location, maintenance and fuel state, and so on.

In MOPS, Maintenance requirements for locomotives are defined for each type – how many days between each service and how many hours the loco will be in the shop. The maintenance state – how many hours until a particular loco is due a service, or how long until it is back in traffic – is held for the locomotive. Locomotives enter maintenance by spotting the locomotive at a Loco Maintenance place: at this time, an hourly counter will start, counting down until the locomotive can re-enter traffic. When locomotives need maintenance, then a flash message will be sent to all users.

Locomotives come in different shapes and forms. One of the primary distinctions is that locos can be Steam, Diesel or Electric. There’s a fourth definition – Other – for any other type not covered by the first three. Apart from Electric locos, all locomotives require refuelling on a regular basis – the fuel capacity, and consumption rate, are defined against the Type, while the amount of fuel left is held against individual locomotives. Simply being on a train uses up fuel. Making ‘corrective’ moves in MOPS also uses up fuel. Fuel is restored by spotting a locomotive at an appropriate refuelling place. When a locomotive is low on fuel then a Flash message will be sent to all users.

As well as having locomotives capable of making their own way in the world, there are additional operating modes for locomotives. A slave, or dummy, unit can’t pull a train, as it will have no cab from which to operate. A Multiple Unit Set can have unpowered (or other powered) rolling stock attached to it, after which it will be treated as a single unit (until it’s broken up again). There’s more about making Multiple Unit Sets later in this guide.

Locomotives can also be powered or unpowered. When powered, it will be capable of hauling cars and it will have an effective weight of zero. When unpowered, it will have no haulage capability but will have an effective weight.

Finally, locomotives can be allocated to a Home Station; it’s a place where Locomotives should normally go when there is no work for them: typically this would probably be where they get their maintenance carried out.

There are a range of commands to create and then look after locomotives in MOPS. Some of the commands will be covered later in this Guide, where normal operational running is discussed, but in the meantime, create the size of the code fields that are required for both Locomotive Types and Locomotives:

```
>chparm loctsize;6
>chparm locosize;5
```

and log off then log on again to reload the size parameters.

To add a new Locomotive Type, you need to include the following details (the data in brackets refer to the example immediately following):

- a code for the locomotive type (GF-70X);
- a name for the locomotive type (GENERAL FREIGHT TYPE 70);
- whether it's Diesel, Electric, Steam or Other (D)
- the tonnage (or number of cars, or other unit) it's capable of hauling(5400)
- fuel capacity (3200)
- rate at which fuel is consumed when hauling a train (34)
- the days between maintenance (125)
- the time in works when in maintenance (48)
- the weight and length of the locomotive type, and (430) and (68)
- whether it operates on its own, is part of a multiple set (with non-powered vehicles) or whether it's a slave/ dummy loco (can't be allocated to a train on its own) (I)

and then added to MOPS:

```
>adloct GF-70X;General Freight Type 70;d;5400;3200;34;125;48;430;68;i
14:56 ADD LOCOMOTIVE TYPE BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW LOCOMOTIVE TYPE ADDED SUCCESFULLY
LOCOMOTIVE TYPE:GF-70X GENERAL FREIGHT TYPE 70 DIESEL LOCOMOTIVE
HAUL:5400  WT:430 LGTH:68 FUEL CAP:3200 RATE:34
MAINT INTERVAL:125 WORKS TIME:48 (LOCOMOTIVE CAN OPERATE INDEPENDENTLY)
```

Locomotives can now be added to MOPS: some of the information is defaulted for locomotives (for example, it's assumed that the locomotive starts with a full fuel load). All that's needed for each Locomotive is its type, the owning Railroad and it's Home Station (ie base depot). So allocating a new locomotive #21566:

```
>adloco 21566;gf-70x;apr;2100
14:56 ADD LOCOMOTIVE BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW LOCOMOTIVE ADDED SUCCESFULLY
21566 (GF-70X GENERAL FREIGHT TYPE 70)
APR ATLANTIC AND PACIFIC RAILROAD  HOME STATION:2100 RUSSEL ROAD
FUEL LOAD:3200 MAINT DUE:125DAYS
```

and that entry puts us in the Haulage business.

Car Classes, Car Types and Cars

With Locomotives, we set up a Type of Locomotive and then created individual Locomotives that were of that type. Cars are a little different.

At a very general level, Cars are identified as a type of Class. These classes are a very high level of identifier: typically identifying whether a car is a box car, hopper or flat. These Classes can be set up as required to suit individual railroads, and gives MOPS the ability to, for example, ‘order 4 box cars’ without having to specify the exact car type details.
Each Class is then sub-divided into Car Types – and it’s the Car Types that contain information about the length, height, capacity about types of cars. This equates to the Locomotive Type records.

Then each car is attached to a Car Type and, by default, attached to a Car Class.
At the Car Class level, all that’s required to create the record is a code and the Class Name.
The Car Type requires a lot more detail. In addition to being attached to a Car Class, it needs a name, length and unladen weight. It also needs a capacity, which determines how much is loaded into the wagon so it’s important that the capacity is associated with the production rate of industries that are associated with those types. Note that the laden weight of a car is a combination of its unladen weight and its capacity.

The Operating Mode indicates whether it can operate as a car on its own or whether it’s operating as part of a Multiple Unit set (it can then be linked to a Multiple Unit power unit – see later).

Finally, the loading and unloading codes can be added. This determines what mechanisms need to be in place to load and unload goods. Loading and Unloading codes need to agree to allow goods to be loaded and unloaded:

Loading can happen if:	**Unloading can happen if:**
Commodity Loading code agrees with...	
the Loading code for the Industry...	The Unloading code for the Industry...
and the Loading code for the Car	agrees with the Unloading Code for the Car

Then the individual cars can be set up. As before, the size of the three code fields for each of Car Class, Car Type and Car can be set by the user:

```
>chparm classize;3
>chparm cartsize;4
>chparm carxsize;7
```

And, as usual, log off MOPS and log back on again.

Then set up the Car Class:

```
>adclas flt;flat car
14:56 ADD CAR CLASSIFICATION BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW CAR CLASS ADDED SUCCESFULLY
FLT FLAT CAR
```

Then the Car Type:

```
>adcart f36a;flat car type 36/a;55;125;34;i;ramp;ramp;flt
14:56 ADD CAR TYPE BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW CAR TYPE ADDED SUCCESFULLY
F36A FLAT CAR TYPE 36/A FLT FLAT CAR INDEPENDENT
LENGTH:55 CAPACITY:125 UNLADEN WT:34
LOADING:RAMP (LOADING PLATFORM) UNLOADING:RAMP (LOADING PLATFORM)
```

Which creates a new Car Type named F36A with information about its weight, length and capacity. This new type is loaded and unloaded via a RAMP.

Finally, we can set up a Car and, like locomotives earlier, we set it up with a running number, attach it to a Car Type (it will automatically link it to Car Class) and associated it with a Railroad and Home Station.

```
>adcarx 0005048;f36a;prr;1000
14:56 ADD CAR DETAIL BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NO DEFAULT VALUE SET FOR CAR MAINTENANCE, 90 DAYS ASSUMED
NEW CAR ADDED SUCCESFULLY
RUNNING NUMBER:0005048 TYPE:F36A FLAT CAR TYPE 36/A CLASS:FLT
PRR THE PITTSFIELD RAILROAD  HOME STATION:1000 SPRINGFIELD
AT:1000 SPRINGFIELD
```

One of the bits of information indicated in the output is that there is no default value for maintenance. For Cars, maintenance is set for all cars on MOPS – there’s no ability to set maintenance for individual Car Types or Car Classes (generally, cars will need a lot less maintenance than locomotives). If the default time of 90 days isn’t appropriate, set up the required parameters for the time between maintenance for cars, and the time in maintenance:

```
>chparm carmaint;120
14:56 CHANGE PARAMETER BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
CAR MAINTENANCE INTERVAL (CARMAINT) CHANGED TO 120

>chparm carworks;55
14:56 CHANGE PARAMETER BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
CAR MAINTENANCE TIME (CARWORKS) CHANGED TO 55
```

There’s an additional marker that can be created to specifically identify Passenger Cars. In a couple of the queries that will be seen later on, it’s possible to distinguish between passenger cars and freight cars. All Passenger Cars need to be identified with a single Car Class code, then it’s simply a case of setting this code up on the parameter file:

```
>chparm passenger;pas
12:10 CHANGE PARAMETER BY BF ON 27JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
PASSENGER CAR CLASS (PASSENGER) CHANGED TO PAS
```

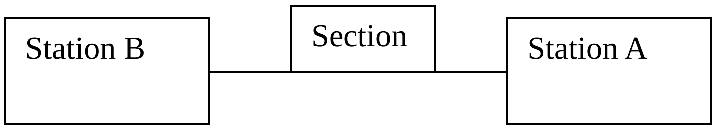
However, it is still necessary to set up a loading code for passengers to enable car types to be created – just treat passengers as *self-loading perishable freight*.

Routes and Timetables

Routes and Sections

MOPS understands routes, timetables and trains. They are all inter-linked – a train follows a timetable, which is a set of times against a route.

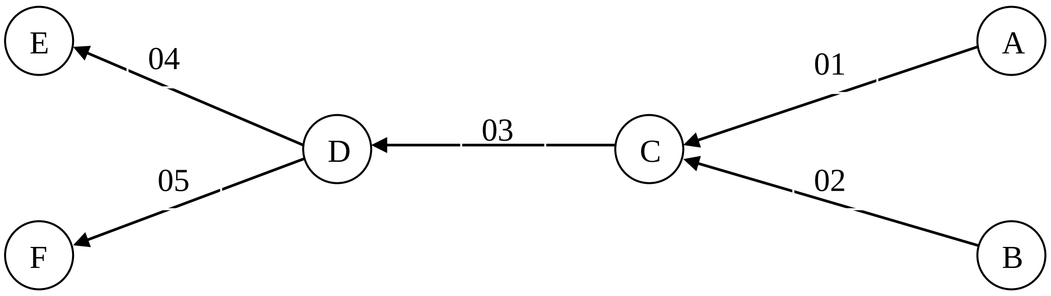
From a MOPS point of view, a Route consists of a series of Sections, with a section being defined as a Station to another, adjacent Station:



A Route, therefore, consists of a series of contiguous sections linked together to form a single, unbroken path. Routes should be set up for every expected or planned journey from one destination to another: a reverse route isn't required as schedules can be added running in the opposite direction to the original route.

There are instances where Routes may merge, have a common number of sections, then move apart again. In this instance, although sections may be shared, they would be defined as separate routes.

Although this requires a little more setting up, it does increase ease of use in the normal running of the system, especially in timetabling or running trains. An example of a set of routes, over a short line, is shown below. Stations are labelled as A, B, C etc; sections are labelled as 01, 02 etc, and routes defined as R1, R2, and so on.



As a complete exercise, we set up MOPS with six stations, A through F:

```
ARE STAT SHORT NA LONG NAME===================== TYP ALIA
SCH A    STAX_A   STATION A                      STN
SCH B    STAX_B   STATION B                      STN
SCH C    STAX_C   STATION C                      STN
SCH D    STAX_D   STATION D                      STN
SCH E    STAX_E   STATION E                      STN
SCH F    STAX_F   STATION F                      STN
```

MOPS understands one Primary direction, and all routes have to either be in that direction or the opposite direction (including Up/Down for UK users). The Railroad default direction is set up on the parameter file as a DIRECTION parameter, and teh value must be one of W, E, S, N, U or D.

Next, we set up the routes that we need. Generally, not all possible routes are set up; instead, only the routes that are most likely to be used are created – other routes can be added as required at a later date. For the moment, we will set up routes for A-E, A-F and B-E. Note that routes are set up as origin-destination, even if there are common parts of the routes.

We don't have to worry about the reverse directions: when we set up a route, it's defined with a direction fro the parameter file. When we set up a Schedule we can have it run with, or opposite to, the default direction.

Let's set up the three Routes we require using the ADROUT command. To see what information you need to put in for the ADROUT command, put in ADROUT followed by a question mark (ie **ADROUT ?**) and MOPS will let you know what data it is expecting. For more help, see the footnote![1]

The command is reasonably straightforward, and it's worthwhile getting to know how to get the prompts for MOPS commands and also starting to use the reference part of the guide - so in this case we will leave you to work this one out!

What is important to know at this stage is that a Schedule is based directly on a route – including origin and destination points. So a schedule that runs only part of *another* route needs its **own** route to be defined. In the example, if we want a schedule to run ACDE, and another one to run only ACD, then we need to set up two routes on which we base the schedules.

Think of the routes as 'templates' for the schedules, rather than physical routes themselves.

```
RO NAME========================  DIR
STATUS DEPART======= ARRIVE=======
R1 ROUTE 1                        W  INCMPL
R2 ROUTE 2                        W  INCMPL
R3 ROUTE 3                        W  INCMPL
```

The three routes are set up with a default direction of West (the Status is incomplete because the route's not ready to put a Schedule on to).

The sections are now added for each route. Route 1 consists of stations A-C, then C-D, then D-E. The sections are actually added as links of stations, and it's important that the stations are in order and are contiguous. The order is given by a section reference (which is unique to the route), which is used to sort the sections in order. Sections are added using the ADSECT command.

```
>adsect r1;01;a;c
14:56 ADD ROUTE SECTION BY CC ON 24JUN1972
NEW ROUTE SECTION ADDED SUCCESFULLY
ROUTE: R1 FROM: A STATION A TO: C STATION C
```

[1] ADROUT ? will show you:
```
ADROUT route;route name
```

So to enter the example shown as Route 1 (R1), enter:
```
>adrout r1;route 1
12:00 ADD ROUTE BY DEFAULT ON 25JUN1972
NEW ROUTE HEADER ADDED SUCCESFULLY
R1 ROUTE 1  DEFAULT DIRECTION: W
```

And if we do this for all the sections on route 1:

```
ROUTE:R1 ROUTE 1 DIRN:NOT KNOWN STATUS:INCOMPLETE
ID=== SECTION= DEPARTING==== ARRIVING=====
    1        1 A STAX_A      C STAX_C
    2        3 C STAX_C      D STAX_D
    3        4 D STAX_D      E STAX_E
```

And setting up the remaining routes in a similar way we get:

```
RO NAME=========================  DIR STATUS DEPART======= ARRIVE=======
R1 ROUTE 1                         W  INCMPL A STAX_A      E STAX_E
R2 ROUTE 2                         W  INCMPL A STAX_A      F STAX_F
R3 ROUTE 3                         W  INCMPL B STAX_B      E STAX_E
```

And finally we need to publish the routes to make them available so that we can create Schedules. This is a two-step process – first the route is validated (MOPS makes sure that the sections are contiguous and that they are in order) using the command VALIDR. After validation, they are made available by Publishing the route with the PUBLSH command:

```
RO NAME=========================  DIR STATUS DEPART======= ARRIVE=======
R1 ROUTE 1                         W  PUBLSH A STAX_A      E STAX_E
R2 ROUTE 2                         W  INCMPL A STAX_A      F STAX_F
R3 ROUTE 3                         W  INCMPL B STAX_B      E STAX_E
```

Schedules and Timings

Having set up the Routes, we now need to set up the Schedules. Schedules consist of two elements:

- The Schedules themselves are the 'header' records for the timetable – these tell us when train is due to leave the originating station, which direction it is travelling in, and so on.
- The Timings are simply the times at Stations along the route for that schedule.

Setting up a Schedule, we need to know the Route, the Class (or the Priority) of the Schedule, on what days the Schedule operate (you can select any combination of weekday, and also include a holiday option) and the direction (which must be equal to or opposite the direction set up by the route).

```
adschd #1;r1;the alphabet flier;1;1.3.56.8;w
14:56 ADD SCHEDULE BY CC ON 24JUN1972
SCHEDULE ADDED FOR ROUTE R1 ROUTE 1
#1 THE ALPHABET FLIER WESTBOUND CLASS:1 STATUS:INACTIVE
RUNS: MONDAY WEDNESDAY FRIDAY SATURDAY HOLIDAYS
FROM:A STATION A  TO:E STATION E
```

Working from left to right, we've added the route (R1), the name, the Class (1), the days of the week on which it runs, and the direction. The days on which it runs is given by numbers: If it runs on a Monday, enter 1 as the first character. If it doesn't run, enter a dot.

If it runs on a Tuesday, enter the second character as a 2. If it doesn't run, enter a dot.

For the eight character (MOPS expects eight characters), enter an 8 in the eight position if it runs on Holidays, and a dot if it doesn't. This Holiday option is linked to the Holidays declared in the MOPS calendar, so it's possible to throw in some holiday schedules as a variant. In this example:

```
1.3.56.8
```

Position 1 = 1 so it runs on a Monday; position 2 is a dot so it doesn't run; position 3 is a three so it runs on a Wednesday, and so forth.

The final piece of information is the direction. If the direction of the schedule equals the direction of the route, then the schedule will follow the route. If the direction is opposite, the schedule will reverse the route. Routes/Schedules deal with opposites as follows:

West is opposite to East
North is opposite to South
Up is opposite to Down (Up/Down are used in the UK, and relate to where London is)

```
SC RO NAME========================= STATUS== RUN DAYS D C =DEP ORIG DEST
#1 R1 THE ALPHABET FLIER            INACTIVE 1.3.56.8 W 1      A    E
#2 R1 THE EASTBOUND FLIER           INACTIVE 1.3.56.8 E 1      E    A
```

There are a number of different mechanisms to add a schedule – using the Prompt method, the Edit method, and the Copy method. The Copy method can only be used to copy an existing schedule and create a new start time.

The Prompt method is started with the ADTIMS command. Normally, in MOPS, all the data is entered and then validated before being either accepted or rejected. This command is different: MOPS will ask for information to be entered a station at a time for the Schedule. The first piece of information that is required is the departure time from the origin:

```
>adtims #1
14:56 ADD SCHEDULE TIMINGS BY CC ON 24JUN1972
SCHEDULE ENTRY MODE: ENTER TIME HHMM OR <X> TO QUIT
TIME DEPARTING A STAX_A >
```

And a time can be entered for when the train is due to leave. MOPS then prompts for the arrival time at the next station, and the departure time, and so on until the information is complete:

```
>adtims #1
14:56 ADD SCHEDULE TIMINGS BY CC ON 24JUN1972
SCHEDULE ENTRY MODE: ENTER TIME HHMM OR <X> TO QUIT
TIME DEPARTING A STAX_A >1000
TIME ARRIVING  C STAX_C >1250
TIME DEPARTING C STAX_C >1425
TIME ARRIVING  D STAX_D >1600
TIME DEPARTING D STAX_D >1600
TIME ARRIVING  E STAX_E >1832
UPDATE OF SCHEDULE TIMINGS FOR #1 COMPLETED
```

And the results can be seen with the TIMING command:

```
>timing #1
14:56 LIST TIMINGS FOR SELECTED SCHEDULE BY CC ON 24JUN1972
SCHEDULE: #1 THE ALPHABET FLIER  (SCHEDULE STATUS:INACTIVE)
       DIRECTION: NOT KNOWN

STAT NAME==== TYP =ARR =DEP INSTRUCTIONS =========================
A    STAX_A   STN      1000
C    STAX_C   STN 1250 1425
D    STAX_D   STN 1600 1600
E    STAX_E   STN 1832
```

Note that this mechanism cannot be used if the batch, or automated, set up is used – use the Edit method instead. Note also that MOPS does handle cross-midnight trains with a degree of tolerance – basically, it accepts a train time after 9pm followed by a train time before 3am.

The Edit method uses the more usual command approach to putting data into MOPS. The command is CHTIMS, and it's simply a case of putting in an arrival and departure time for each section of the route one at a time. Use **`CHTIMS ?`**[2] to see what data is needed for the command.

```
>chtims #2;4;1200;1426
14:56 CHANGE SCHEDULE TIMINGS BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
SCHEDULE TIMINGS CHANGED FOR:#2 E:1200 D:1426

>chtims #2;3;1426;1650
14:56 CHANGE SCHEDULE TIMINGS BY CC ON 24JUN1972
SCHEDULE TIMINGS CHANGED FOR:#2 D:1426 C:1650

>chtims #2;1;1815;2000
14:56 CHANGE SCHEDULE TIMINGS BY CC ON 24JUN1972
SCHEDULE TIMINGS CHANGED FOR:#2 C:1815 A:2000

>timing #2
14:56 LIST TIMINGS FOR SELECTED SCHEDULE BY CC ON 24JUN1972
SCHEDULE: #2 THE EASTBOUND FLIER  (SCHEDULE STATUS:INACTIVE)
       DIRECTION: EAST

STAT NAME==== TYP =ARR =DEP INSTRUCTIONS =========================
E    STAX_E   STN      1200
D    STAX_D   STN 1426 1426
C    STAX_C   STN 1650 1815
A    STAX_A   STN 2000
 ** END OF DATA:3  RECORDS DISPLAYED **
```

This mechanism can also be used to correct or amend information entered through the Prompt method.

The final mechanism for entering data is the **Copy** method. This simply takes an existing schedule and creates an exact copy, but offsetting the times by the proposed new start time:

```
>cpschd #1;#3;the westbounder;1800
14:56 COPY SCHEDULE BY CC ON 24JUN1972
NEW SCHEDULE CREATED #3

>lischd
14:56 LIST ALL SCHEDULES BY CC ON 24JUN1972
SC RO NAME========================= STATUS== RUN DAYS D C =DEP ORIG DEST
#1 R1 THE ALPHABET FLIER            INACTIVE 1.3.56.8 W 1 1000 A    E
#2 R1 THE EASTBOUND FLIER           INACTIVE 1.3.56.8 E 1 1200 E    A
#3 R1 THE WESTBOUNDER               INACTIVE 1.3.56.8 W 1 1800 A    E
 ** END OF DATA:3  RECORDS DISPLAYED **

>timing #3
14:56 LIST TIMINGS FOR SELECTED SCHEDULE BY CC ON 24JUN1972
SCHEDULE: #3 THE WESTBOUNDER  (SCHEDULE STATUS:INACTIVE)
       DIRECTION: WEST

STAT NAME==== TYP =ARR =DEP INSTRUCTIONS =========================
A    STAX_A   STN      1800
C    STAX_C   STN 2050 2225
D    STAX_D   STN 0000 0000
E    STAX_E   STN 0232
 ** END OF DATA:3  RECORDS DISPLAYED **
```

[2] CHTIMS schedule;section;depart;arrive

The last thing that needs to be done is to make the schedule available for running trains:

```
>active #3
18:00 ACTIVATE SCHEDULE BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
SCHEDULE #3 SET TO ACTIVE STATUS: AVAILABLE FOR TRAIN RUNNING

>lischd
18:00 LIST ALL SCHEDULES BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
SC RO NAME============================ STATUS== RUN DAYS D C =DEP ORIG DEST
#1 R1 THE ALPHABET FLIER               INACTIVE 1.3.56.8 W 1 1000 A    E
#2 R1 THE EASTBOUND FLIER              INACTIVE 1.3.56.8 E 1 1200 E    A
#3 R1 THE WESTBOUNDER                  ACTIVE   1.3.56.8 W 1 1800 A    E
```

Full Worked Example of a Route and Schedule.

This is a full, worked example of a route from one station to another. The stations used are those on the example file, and you can see the route on the map distributed with MOPS.

The worked example shows all the inputs and outputs from setting up a route from Avebury Power to Stockbridge Heights, the sections entered, and four schedules added, with timings, (two schedules in each direction).

First the Route:

```
> ADROUT 08;AVEBURY-S/BRIDGE HEIGHTS;S
NEW ROUTE HEADER ADDED SUCCESFULLY
08 AVEBURY-S/BRIDGE HEIGHTS  DEFAULT DIRECTION: S
```

Then the Sections:

```
> ADSECT 08;13;2060;2050
NEW ROUTE SECTION ADDED SUCCESFULLY
ROUTE: 08 FROM: 2060 AVEBURY ROAD PWR STN TO: 2050 AVEBURY ROAD JCN

> ADSECT 08;14;2050;2000
NEW ROUTE SECTION ADDED SUCCESFULLY
ROUTE: 08 FROM: 2050 AVEBURY ROAD JCN TO: 2000 RUSSEL ROAD JUNCTION

> ADSECT 08;15;2000;3000
NEW ROUTE SECTION ADDED SUCCESFULLY
ROUTE: 08 FROM: 2000 RUSSEL ROAD JUNCTION TO: 3000 WESTFIELD

> ADSECT 08;16;3000;4000
NEW ROUTE SECTION ADDED SUCCESFULLY
ROUTE: 08 FROM: 3000 WESTFIELD TO: 4000 STOCKBRIDGE

> ADSECT 08;17;4000;4100
NEW ROUTE SECTION ADDED SUCCESFULLY
ROUTE: 08 FROM: 4000 STOCKBRIDGE TO: 4100 STOCKBRIDGE YARD

> ADSECT 08;18;4100;4200
NEW ROUTE SECTION ADDED SUCCESFULLY
ROUTE: 08 FROM: 4100 STOCKBRIDGE YARD TO: 4200 STOCKBRIDGE JUNCTION

> ADSECT 08;19;4200;4250
NEW ROUTE SECTION ADDED SUCCESFULLY
ROUTE: 08 FROM: 4200 STOCKBRIDGE JUNCTION TO: 4250 STOCKBRIDGE HEIGHTS
```

Then Validate and Publish the Route:

```
> VALIDR 08
VALIDATING ROUTE FOR AVEBURY-S/BRIDGE HEIGHTS
SEQUENCE:13 FROM: 2060 AVEBURYP TO: 2050 AVEBURYP
SEQUENCE:14 FROM: 2050 AVEBURYP TO: 2000 RSLRDJCN
SEQUENCE:15 FROM: 2000 RSLRDJCN TO: 3000 WSTFIELD
SEQUENCE:16 FROM: 3000 WSTFIELD TO: 4000 STCKBRDG
SEQUENCE:17 FROM: 4000 STCKBRDG TO: 4100 STKBRGYD
SEQUENCE:18 FROM: 4100 STKBRGYD TO: 4200 STKBRGJN
SEQUENCE:19 FROM: 4200 STKBRGJN TO: 4250 STKBRGHT
ROUTE VALIDATED: SET TO DRAFT STATUS

> PUBLSH 08
PUBLISHING ROUTE FOR AVEBURY-S/BRIDGE HEIGHTS
ROUTE SET TO PUBLISHED STATUS
```

Add the Four Schedules we want to run (two each way):

```
> ADSCHD #51;08;COAL TRIP #51 MTY   ;1;1234567.;S
SCHEDULE ADDED FOR ROUTE 08 AVEBURY-S/BRIDGE HEIGHTS
#51 COAL TRIP #51 MTY SOUTHBOUND CLASS:1 STATUS:INACTIVE
RUNS: MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY SUNDAY
FROM:2060 AVEBURY ROAD PWR STN  TO:4250 STOCKBRIDGE HEIGHTS

> ADSCHD #52;08;COAL TRIP #52 LOAD  ;1;1234567.;N
SCHEDULE ADDED FOR ROUTE 08 AVEBURY-S/BRIDGE HEIGHTS
#52 COAL TRIP #52 LOAD NORTHBOUND CLASS:1 STATUS:INACTIVE
RUNS: MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY SUNDAY
FROM:4250 STOCKBRIDGE HEIGHTS  TO:2060 AVEBURY ROAD PWR STN

> ADSCHD #53;08;COAL TRIP #53 MTY   ;1;123456.8;S
SCHEDULE ADDED FOR ROUTE 08 AVEBURY-S/BRIDGE HEIGHTS
#53 COAL TRIP #53 MTY SOUTHBOUND CLASS:1 STATUS:INACTIVE
RUNS: MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY HOLIDAYS
FROM:2060 AVEBURY ROAD PWR STN  TO:4250 STOCKBRIDGE HEIGHTS

> ADSCHD #54;08;COAL TRIP #54 LOAD  ;1;123456.8;N
SCHEDULE ADDED FOR ROUTE 08 AVEBURY-S/BRIDGE HEIGHTS
#54 COAL TRIP #54 LOAD NORTHBOUND CLASS:1 STATUS:INACTIVE
RUNS: MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY HOLIDAYS
FROM:4250 STOCKBRIDGE HEIGHTS  TO:2060 AVEBURY ROAD PWR STN
```

Add the timings at each station for the four schedules. The Prompt method could have been used to add the data; or we could have created one in each direction and used the Copy method to create the second:

```
> CHTIMS #51;13;0600;0608
SCHEDULE TIMINGS CHANGED FOR:#51 2060:0600 2050:0608

> CHTIMS #51;14;0608;0620
SCHEDULE TIMINGS CHANGED FOR:#51 2050:0608 2000:0620

> CHTIMS #51;15;0620;0638
SCHEDULE TIMINGS CHANGED FOR:#51 2000:0620 3000:0638

> CHTIMS #51;16;0638;0654
SCHEDULE TIMINGS CHANGED FOR:#51 3000:0638 4000:0654

> CHTIMS #51;17;0654;0702
SCHEDULE TIMINGS CHANGED FOR:#51 4000:0654 4100:0702

> CHTIMS #51;18;0702;0705
SCHEDULE TIMINGS CHANGED FOR:#51 4100:0702 4200:0705

> CHTIMS #51;19;0705;0737
SCHEDULE TIMINGS CHANGED FOR:#51 4200:0705 4250:0737

> CHTIMS #52;19;0832;0858
SCHEDULE TIMINGS CHANGED FOR:#52 4250:0832 4200:0858

> CHTIMS #52;18;0858;0901
SCHEDULE TIMINGS CHANGED FOR:#52 4200:0858 4100:0901

> CHTIMS #52;17;0901;0909
SCHEDULE TIMINGS CHANGED FOR:#52 4100:0901 4000:0909

> CHTIMS #52;16;0909;0925
SCHEDULE TIMINGS CHANGED FOR:#52 4000:0909 3000:0925

> CHTIMS #52;15;0925;0943
SCHEDULE TIMINGS CHANGED FOR:#52 3000:0925 2000:0943
```

```
> CHTIMS #52;14;0943;0955
SCHEDULE TIMINGS CHANGED FOR:#52 2000:0943 2050:0955

> CHTIMS #52;13;0955;1003
SCHEDULE TIMINGS CHANGED FOR:#52 2050:0955 2060:1003

> CHTIMS #53;13;1100;1108
SCHEDULE TIMINGS CHANGED FOR:#53 2060:1100 2050:1108

> CHTIMS #53;14;1108;1120
SCHEDULE TIMINGS CHANGED FOR:#53 2050:1108 2000:1120

> CHTIMS #53;15;1120;1138
SCHEDULE TIMINGS CHANGED FOR:#53 2000:1120 3000:1138

> CHTIMS #53;16;1138;1154
SCHEDULE TIMINGS CHANGED FOR:#53 3000:1138 4000:1154

> CHTIMS #53;17;1154;1202
SCHEDULE TIMINGS CHANGED FOR:#53 4000:1154 4100:1202

> CHTIMS #53;18;1202;1205
SCHEDULE TIMINGS CHANGED FOR:#53 4100:1202 4200:1205

> CHTIMS #53;19;1205;1237
SCHEDULE TIMINGS CHANGED FOR:#53 4200:1205 4250:1237

> CHTIMS #54;19;1332;1358
SCHEDULE TIMINGS CHANGED FOR:#54 4250:1332 4200:1358

> CHTIMS #54;18;1358;1401
SCHEDULE TIMINGS CHANGED FOR:#54 4200:1358 4100:1401

> CHTIMS #54;17;1401;1409
SCHEDULE TIMINGS CHANGED FOR:#54 4100:1401 4000:1409

> CHTIMS #54;16;1409;1425
SCHEDULE TIMINGS CHANGED FOR:#54 4000:1409 3000:1425

> CHTIMS #54;15;1425;1443
SCHEDULE TIMINGS CHANGED FOR:#54 3000:1425 2000:1443

> CHTIMS #54;14;1443;1455
SCHEDULE TIMINGS CHANGED FOR:#54 2000:1443 2050:1455

> CHTIMS #54;13;1455;1503
SCHEDULE TIMINGS CHANGED FOR:#54 2050:1455 2060:1503
```

And finally set the Schedules to Active:

```
> ACTIVE #51
SCHEDULE #51 SET TO ACTIVE STATUS: AVAILABLE FOR TRAIN RUNNING

> ACTIVE #52
SCHEDULE #52 SET TO ACTIVE STATUS: AVAILABLE FOR TRAIN RUNNING

> ACTIVE #53
SCHEDULE #53 SET TO ACTIVE STATUS: AVAILABLE FOR TRAIN RUNNING

> ACTIVE #54
SCHEDULE #54 SET TO ACTIVE STATUS: AVAILABLE FOR TRAIN RUNNING
```

And then it’s all done. Although the schedules take a little planning (just like the real world) actually setting them up on MOPS is reasonably straightforward.

Instructions

The final bits of data that can be added up in advance of operations are any specific instructions. These can be added at the Station, Route or Schedule level, and are displayed on output at appropriate times, and are simply bits of operational information that may be required at appropriate places – for example, a Station may have special operating requirements, a Route may have special speed restrictions in place or a Schedule may have additional information. In the example below, notice the additional information on the Timing enquiry about sleeping cars and Eastbound trains.

```
>adinst schd;#3;convey sleeping cars from station C
14:56 ADD INSTRUCTION BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW INSTRUCTION ADDED SUCCESFULLY
SCHD #3 CONVEY SLEEPING CARS FROM STATION C

>adinst stax;d;no access to road 1 for eastbound trains
14:56 ADD INSTRUCTION BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
NEW INSTRUCTION ADDED SUCCESFULLY
STAX D NO ACCESS TO ROAD 1 FOR EASTBOUND TRAINS

>timing #3
14:56 LIST TIMINGS FOR SELECTED SCHEDULE BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
SCHEDULE: #3 THE WESTBOUNDER  (SCHEDULE STATUS:INACTIVE)
       DIRECTION: NOT KNOWN
 -  CONVEY SLEEPING CARS FROM STATION C

STAT NAME==== TYP =ARR =DEP INSTRUCTIONS =========================
A    STAX_A   STN      1800
C    STAX_C   STN 2050 2225
D    STAX_D   STN 0000 0000 NO ACCESS TO ROAD 1 FOR EASTBOUND TRAINS
E    STAX_E   STN 0232
 ** END OF DATA:3  RECORDS DISPLAYED **
```

Automating Set-up

It is possible to automate setup and to create the required data away from MOPS, then have MOPS load the data as a batch file. The loading uses the same commands as a user, but simply processes them one after another. The batch file should be set up in a plain text editor; MOPS will not handle formatting codes. Comments can be added by adding dashes and an arrow at the start of the line, as shown below. Trailing spaces on text are removed. A mix of data is allowed, and in any order.

```
-----> add default station data --------------------------------------
adstax 1000;sprnglfd;springfield         ;spr;sta;
adstax 1050;sprfldyd;springfield yard    ;spr;sta;
adstax 1100;hartford;hartford            ;spr;sta;1000
adstax 1200;boston  ;boston              ;spr;sta;1000
adstax 1300;provdnce;providence          ;spr;sta;1000
adstax 2000;rslrdjcn;russel road junction;stk;jcn;
adstax 2100;rsslroad;russel road         ;stk;stn;
adstax 3000;wstfield;westfield           ;stk;stn;
adstax 4000;stckbrdg;stockbridge         ;stk;stn;
adstax 4100;stkbrgyd;stockbridge yard    ;stk;stn;
adstax 4200;stkbrgjn;stockbridge junction;stk;jcn;
adstax 4250;stkbrght;stockbridge heights ;stk;stn;
adstax 5000;felhamjn;felham junction     ;stk;jcn;
```

To run the batch file, place the file with a ‘.txt’ suffix in the same directory as the database. Then restart MOPS and start MOPS with option 3. MOPS will prompt for a filename:

```
MOPS NOT STARTED.  PLEASE SELECT:
    1 Start Background tasks for MOPS on this console
    2 Enter MOPS in Maintenance Mode (no background tasks)
    3 Run batch file load (.TXT from data directory
    4 Create Calendar for 5-year period (1950/1955/1960.../2010)
    9 Quit (this will terminate this session)
==> 3
ENTER FILENAME > ds1
```

Once you’ve entered the filename, press Enter. If the file can’t be found, then nothing will be processed. If the file is good, then it will attempt to process everything on the file. A lot of information will flow past on the screen, eventually returning to menu:

```
...>>> ADPLAX 6100;98;VALLEY YARD CAR CLEAN    ;C;
NEW PLACE ADDED SUCCESFULLY
STATION:6100 VALLEY YARD  PLACE:98 VALLEY YARD CAR CLEAN (GENERATED ID IS 9)
TRACK LENGTH:0 CLEAN CARS
>>>
>>> ADPLAX 1050;75;SPRINGFIELD YARD ARRIVALS;X;4000
NEW PLACE ADDED SUCCESFULLY
STATION:1050 SPRINGFIELD YARD  PLACE:75 SPRINGFIELD YARD ARRIVALS (GENERATED
ID IS 10)
TRACK LENGTH:4000
>>>
>>> ADPLAX 1050;76;SPRINGFIELD YARD HOLDING ;X;12500
NEW PLACE ADDED SUCCESFULLY
STATION:1050 SPRINGFIELD YARD  PLACE:76 SPRINGFIELD YARD HOLDING (GENERATED
ID IS 11)
TRACK LENGTH:12500
>>>
>>> ADPLAX 1050;77;SPRINGFIELD YARD DEPARTS ;X;4500
NEW PLACE ADDED SUCCESFULLY...
```

Starting a Session

Starting a Session is straightforward. Start MOPS as normal, then sign on (as a supervisor). When the menu appears (assuming MOPS is not started), select option 1 to start the MOPS background task.

```
***************************************************************************
* MOPS running in Maintenance Mode.  No background tasks are running.      *
***************************************************************************

MOPS NOT STARTED.  PLEASE SELECT:
    1 Start Background tasks for MOPS on this console
    2 Enter MOPS in Maintenance Mode (no background tasks)
    3 Run batch file load (.TXT from data directory
    4 Create Calendar for 5-year period (1950/1955/1960.../2010)
    9 Quit (this will terminate this session)
==>
```

You will then see a message indicating that the background tasks have been started, followed by the current MOPS date and time.

```
>>> MOPS BACKGROUND PROCESSES STARTED <<<
... background task running on this console ...
this session will stop when MOPS  is terminated

24JUN1972 14:56
```

There will appear to be no activity for a few minutes, depending on clock speed, the time of MOPS in relation to the hour and so forth. Every so often, messages will appear on the monitor, indicating that MOPS has executed a background task or other event has taken place:

```
24JUN1972 14:56
24JUN1972 15:00 Z006: EXPIRING FLASH MESSAGES
24JUN1972 15:00 Z018: SCHEDULES SET TO ACTIVE
24JUN1972 15:18 Z001: ALBANYCHEM HAS RAISED REQUIREMENT FOR 2 BOX EMPTIES

                 *********************************
                 *** DO NOT CLOSE THIS CONSOLE ***
                 *********************************
```

This session should now be left alone (if you want to see what the background tasks are doing), or minimised onto the task bar. DO NOT CLOSE THE WINDOW – closing the window will case all the background tasks to stop! If the session is closed by mistake, then the background tasks will stop; however, MOPS thinks they are still running. To rest MOPS, open another MOPS session and, after signing on with option 2, type in XXSTOP to reset the database.

You can have multiple sessions of MOPS on the same machine, in other words you can run MOPS as a background task and have another session open for operations on the same PC.

Users that are not supervisors cannot log on to MOPS unless MOPS is running in background mode.

Shutting a session down

Using a session that's logged on as a supervisor, the background tasks can be stopped with XXSTOP. This will close the background session and stop the MOPS clock. The session that’s running the background tasks will automatically close:

```
>xxstop
15:40 STOP MOPS BY CC ON 24JUN1972
>>> MOPS BACKGROUND PROCESSES STOPPED <<<
```

No further activity will be generated by the background tasks, the background tasks session will close. **Note that ANY users that are not supervisors will be thrown off MOPS.**

Operational Processes

Not all commands are covered in this section: a full review of the Commands Reference is recommended. Altogether there are more than 200 commands available in MOPS, and each command is described in the reference section.

How you operate your railroad is up to you. MOPS doesn't 'drive' operations, it simply records what has been planned (through a schedule) or what has happened (warehouse production, car and train movements).

Having started the background tasks, start a second session for inputting commands and requesting lists. In the meantime, the background task session may look as if it's doing nothing for a while, as production spins up in the warehouses. If there are no events to report, the background process will only change once each MOPS hour.

To speed up production processes to start with, set the clock speed to a faster rate (eg CSPEED 8). Alternatively, use the time to sort out the other work on the railroad!

The first alerts that are likely to be seen are requests for empty cars to be delivered to specific industries. We can see current requirements for empty cars with the LEMPTY command:

```
>lempty
15:48 LIST EMPTY CAR REQUESTS BY CC ON 24JUN1972
ORDER INDUSTRY== STATION= COM CARS= CLN DESTINATN= DEST STN REQUESTED======
    1 ALLENHAYES SYRACUSE 420      1   N PITTSFLD_H PITTSFLD 24JUN1972 14:38
    2 EMOUNTAIN  SCHODACK 500      1   N FELHAMCAN  FELHAMJN 24JUN1972 14:44
    3 PROVIDRAMP PROVDNCE 320      1   N FELHAMCAN  FELHAMJN 24JUN1972 14:44
    4 STOCKBWOOD STKBRGHT 300      1   N PROVIDHOSE PROVDNCE 24JUN1972 14:44
    5 MOUNTAINAL RSSLROAD 600      3   N FELHAMCAN  FELHAMJN 24JUN1972 14:47
```

And we can see right away that we have a number of empty cars to supply. Let's have a look at the second one – EMOUNTAIN. By putting in a LIWARE command, we can see that EMOUNTAIN requires BOX cars and that they're at Station 7000. What empty cars do we have available to fulfil this requirement?

The LMTCAR command gives us a list of cars not currently assigned to any work duty:

```
>lmtcar
16:25 REPORT UNALLOCATED EMPTY CARS BY CC ON 24JUN1972
RAI ARE STAT NAME==== PL TYPE CAR==== C BLOCK    LOAD UNLO
PRR SPR 1000 SPRNGFLD    TR50 0005042            RAMP RAMP
PRR SPR 1000 SPRNGFLD    TR50 0005043            RAMP RAMP
PRR SPR 1000 SPRNGFLD    TR50 0005044            RAMP RAMP
PRR SPR 1000 SPRNGFLD    TR50 0005045            RAMP RAMP
PRR SPR 1000 SPRNGFLD    TR50 0005046            RAMP RAMP
```

Other listings, such as LACARS is also available to allow selection of the car that is required. So we can assign one of the cars to fulfil that order –

```
>mtyord 2;0005042
17:24 ALLOCATE EMPTY TO ORDER BY CC ON 24JUN1972
CARS ATTACHED TO ORDER 2 FOR EMOUNTAIN ROCHESTR
CAR: 0005042 CURRENTLY AT: 1000
EMPTY CAR ORDER FULFILLED FOR ORDER 2
```

The next step is to find a train that's going there. The Car's at Station 1000 SPRINGFIED and we need to get it to SCHODACK, which is at Station 7000. We will look for an appropriate train in a moment.

Let's turn our attention now to setting up a train. There are three ways of setting up trains in MOPS – we can run a Scheduled, an Extra train (which follows the same route as a Scheduled train, but on a slightly different timetable) and an Unscheduled Rain (which doesn't have a timetable at all, it makes it up as it goes along).

In a previous section, we went through the process of setting up a Schedule. There's an active schedule (number 23, *The Evening Run*) which runs from Springfield to Schodack. It's due to leave at 1830, so we'll make this the train we are going to run, place the car we are going to move onto it, and run the train.

```
>lischd
18:08 LIST ALL SCHEDULES BY CC ON 24JUN1972
SC RO NAME============================ STATUS== RUN DAYS D C =DEP ORIG DEST
#4 02 EASTERN LOCAL (WEST)             INACTIVE 1...5..8 W 3      7000 2100
22 03 SPRINGFIELD-SCHODACK             ACTIVE   12345678 E 2 0510 1000 7000
#1 R1 THE ALPHABET FLIER               INACTIVE 1.3.56.8 W 1 1000 A    E
#2 R1 THE EASTBOUND FLIER              INACTIVE 1.3.56.8 E 1 1200 E    A
#3 R1 THE WESTBOUNDER                  ACTIVE   1.3.56.8 W 1 1800 A    E
23 03 EVENING RUN                      ACTIVE   12345678 E 2 1830 1000 7000
```

The schedule becomes a train when we attach a locomotive – let's see what power we have available at Springfield:

```
>lsloco ;;;1000
18:46 LIST LOCOMOTIVE DETAILS BY CC ON 24JUN1972
LOCO= TYPE== FUEL= MAINT RPAIR PWR STAT PLACE==================== TRAIN=====
38221 GP38/2  1776    90     0   P 1000                            0
```

There's only one loco available, so we will use this. Match the loco to the schedule:

```
>strain 23;;38221
18:52 SET SCHEDULED TRAIN BY CC ON 24JUN1972
NEW TRAIN: 23  LOCO: 38221 CURRENT STATION: 1000
```

Because the train's now running, we can do a LINEUP enquiry for Schodack to see which trains are on their way:

```
>lineup 7000
18:56 REPORT LINE-UP BY CC ON 24JUN1972
LINE UP ENQUIRY FOR 7000 SCHODACK
TRAIN= DUE= EST= FROM ORIG DEST C #LOCO #CARS #LOAD #MTY= WGHT= LNGTH
23     0435      5000 1000 7000 2     1     0     0     0     0    59
 ** END OF DATA:1  RECORDS DISPLAYED **
```

The next thing to do is to add the car to the train: this would obviously be done down on the track, and reported back that the car is on the train: in MOPS, we add the Car to the Train

```
>acarxt 0005042;23
21:19 ALLOCATE CAR TO TRAIN BY CC ON 24JUN1972
CAR 0005042 ATTACHED TO TRAIN: 23

>lineup 7000
21:19 REPORT LINE-UP BY CC ON 24JUN1972
LINE UP ENQUIRY FOR 7000 SCHODACK
TRAIN= DUE= EST= FROM ORIG DEST C #LOCO #CARS #LOAD #MTY= WGHT= LNGTH
23     0435      5000 1000 7000 2     1     1     0     1    80   109
```

And as you can see from the LINEUP enquiry, the car is now on the train. Let's report the train departing Springfield:

```
        >report 23;dep;1000
        21:25 REPORT TRAIN BY CC ON 24JUN1972
        23 DEPARTED 1000 SPRINGFIELD AT 2125
```

The TRAINS command tells us what trains are running, and where it last reported.

```
        >trains
        21:34 LIST RUNNING TRAINS BY CC ON 24JUN1972
        TRAIN= C TYPE RO SC D ORIG DEST     LAST EXP. ACT.
        23     2 SCHD 03 23 E 1000 7000 DEP 1000 1830 2125
```

We will carry on reporting the train all the way through to Schodack. When we have 'arrived' the train at Schodack, we terminate the train: that releases all the locos and cars and place them at Schodack as a station:

```
        >ttrain 23
        22:07 TERMINATE TRAIN BY CC ON 24JUN1972
        TRAIN 23 TERMINATED AT 7000
```

The car is at 7000 SCHODACK. At some point, the local train will take the train and spot it at EMOUNTAIN. We update MOPS when this is done:

```
        >carxsp 0005042;01
        22:35 SPOT CAR BY CC ON 24JUN1972
        CAR SPOTTED SUCCESFULLY
        0005042 AT:7300 ROCHESTER
        TO MAINT:90 IN MAINT:0
```

And as the car has been spotted at the industry, loading will (in a short time after it has been spotted), begin. After a while, an alert will be issued about a Waybill being raised for the whole process to begin again, this time with a loaded car...

Spotting Cars and Locos is the method of putting vehicles in the correct place for actions to happen; whether it's for loading vehicles (by spotting at an Industry) or for placing vehicles at an appropriate place for maintenance to begin, or for refuelling locos by spotting them at an appropriate refuelling place.

As indicated earlier, it's suggested that a review of the Commands are undertaken; there are a large number of commands and it's not possible to cover all possible options and processes. In addition to this Guide, the Internet site www.railmops.net is available for raising questions and looking for answers.

Correcting Data

There are a number of commands for correcting data in MOPS, and a review of the MOPS Command Reference is recommended for seeing how data can be corrected in MOPS. The most frequent corrections would be for ‘lost’ cars and locos, and changing production information for industries.

If a car is in the wrong location, we can correct it by using the CARXAT command. This takes the car, and places it at the new Station (providing it is not currently on a Train):

```
>carxat 0005043;6200
01:46 LOCATE CAR AT STATION BY CC ON 25JUN1972
LOCATION OF CAR AT STATION CHANGED SUCCESFULLY
0005043 AT:6200 PITTSFIELD
```

A similar command, LOCOAT, exists for relocating a Locomotive. These commands are useful if a car or loco is misplaced, or can be used instead of running trains – simply move the cars with this command instead.

In addition to correcting locations, it is possible to correct fuel levels on locomotives and change maintenance times for locos and cars.

There are some ‘Operational’ data that may also need to be corrected, such as Warehouse Production, Starting trains other than at their origin, terminating trains short of destination and so forth. Although not technically correcting data, these facilities have been included to reflect actions that may take place on a railroad.

Correcting Commands are identified with the words *Correcting Command* at the start of the appropriate command description. Some of them are available to Supervisor users only; others are more generally available depending on the nature of the command.

Tidying MOPS and Backing Up

Backing Up MOPS

MOPS runs from a database called MOPS.db. From the main start-up menu, there's a facility (option 5) to create a back-up copy of MOPS (called MOPS.bak). This is probably worth running at the start of a session, or when there's been an amount of maintenance done on the database, just to ensure that there is a recent copy available that can be returned to in the event of a significant issue. It's probably worth moving that back-up onto another medium.

To restore a database, simply replace MOPS.db with the previously backed-up version.

Purge Print Output files

When MOPS creates printed output, it first creates a file in the directory then actually prints from that. As a result, it is possible to get a large number of files within the MOPS data directory that may no longer be required. To remove these unwanted files, use option 6. Note that this action is irreversible.

Commands Reference

Overview

Virtually all commands in MOPS are six-character commands.

In the procedures outlined below, the steps are described and the command to use to complete the step is identified. There is then a description of the command format, where

- data means that some data must be entered (don't enter the < and >)
- (data) means that entering that data is optional
- CAPS means type it exactly as it is
- [Y/N] means you have a choice of the options shown (in this instance Y or N)
- ^code^ means that you enter a code value. This code must already exist on MOPS.

The format of the command is always the command, then a space, then the data. Different data items are identified by a semi-colon (**never** use a semi-colon in the data itself).

Some data is mandatory; some data is optional; where data is optional it will be identified in the description. Nothing needs to be put into optional fields: but the semi-colon for the field still needs to be entered (so that MOPS 'knows' which field is being skipped). Where data is optional, MOPS will either use the default value or the original value depending on circumstances. If a command has a trailing set of optional fields, and none of these are required, then these can be left out altogether.

For example, using *chstax*:

```
CHSTAX code;(short name);(long name);(area);(station type);(alias)
```

This shows that each field, apart from the Station Code, is optional. If the station long name is changed:

```
chstax 4100;;stockbridge junction
```

The second field, the short name, has been skipped. MOPS knows that you want to retain the value. After the long name there are a further three fields, but as they are not being changed, and as the remaining fields are optional, MOPS knows that the values should not be changed:

```
13:52 CHANGE STATION BY CC ON 24JUN1972
* MOPS running in Maintenance Mode.  No background tasks are running. *
STATION DETAILS CHANGED SUCCESFULLY
4100   STKBRGYD STOCKBRIDGE JUNCTION
AREA: STK STOCKBRIDGE  TYPE: STN STATION
```

For most of the different types of data there are a pair of commands that are not documented. The first is *lx####*, where #### is the short name for the type of data (for example, *lxrail*). This displays the 'raw' data from the database directly to the screen. This is really used for system debugging purposes, but has been retained into the released version. The other command is *px####*, where again #### is the short name for the type of data. This produces an extract file which is suitable for loading into a spreadsheet. These additional commands are shown in the Alphabetical List at the end of this guide.

MOPS Parameters

Calendar: Keeping Track of the Dates

Marking a Holiday *[Supervisor access only]*

This makes MOPS runs schedules on any day where the Holiday marker has been set, rather than running the particular daily schedule for that day. To unset a Holiday, run the command again with the same date – the command acts as a 'toggle' switching the Holiday marker on and off.

```
HOLIDX day;month;year
```

Day – day of month 0-31
Month – three character month, Jan, Feb,...,Dec
Year – Four character year

List Calendar

Displays the MOPS' calendar for the next ten days, showing the day of the week and whether it is a holiday:

```
LICALX

DAY DATE======= HOL
SAT 24 JUN 1972
SUN 25 JUN 1972
MON 26 JUN 1972
TUE 27 JUN 1972
WED 28 JUN 1972 YES
THU 29 JUN 1972
FRI 30 JUN 1972
SAT 01 JUL 1972
SUN 02 JUL 1972
MON 03 JUL 1972
```

Parameters: maintaining data that MOPS needs to run properly

Set the Owning Railroad Name for the system *[Supervisor access only]*

The railroad name is shown on reports and when signing on; it has no other significance. It's really just to show the 'owner' of the system.

```
CHPARM RRNAME;owning railroad
```

Name – name of railroad that is running MOPS. Can be any length. Required.

Set the size of various code fields *[Supervisor access only]*

A number of the code fields can have their size, or length, set to a specific value: so that, for example, a Station code can be defined as being four characters in length.

```
CHPARM [field];size
```

Field – determines the size of the indicated code field. Required.

- **areasize** – size of code field for Areas
- **cartsize** – size of Car Type code field
- **classize** – size of field for Car Class code
- **commsize** – size of field for Commodity Codes
- **curosize** – size of field for Customer Routing code
- **loadsize** – size of field for Loading Codes
- **locosize** – size of field for Locomotive Running numbers
- **loctsize** – size of Loco Type code field
- **plaxsize** – size of field for Place Codes
- **railsize** – size of code field for Railroads
- **routsize** – size of field for Routing code
- **statsize** – size of field for Station Type Codes
- **staxsize** – size of field for Station Codes
- **schdsize** – size of field for Schedule code

Size – size of field. Value must be between 1 and 10. Required.

To set the default Maintenance Times for Cars *[Supervisor access only]*

Cars take their Maintenance schedules from the Parameter file. The two maintenance times are 1) days between service and 2) hours in works.

```
CHPARM setting[CARWORKS/CARMAINT];time
```

Setting – which maintenance time is being set. Required

- Carworks – number of hours in workshop
- Carmaint – number of days between maintenance

Time – value for setting. Required.

To set the internal date for MOPS *[Supervisor access only]*

Sets the internal date and time for MOPS. The date must exist on the Calendar file. The Day of the week is validated as an extra check. *Note that this command does not follow normal formatting procedures for data entry and requires careful entry.*

```
SETTIM ddmmmyy hh:ii:ss dow
```

ddmmmyy – day of month, then month (ie JAN, FEB) the 4-digit year

then a space

then a time as hours, followed by a colon;minutes;colon;seconds

then a space

then the day of week as MON, TUE etc

To set the clock speed for MOPS *[Supervisor access only]*

This changes the internal clock speed on MOPS, so that a fast clock can be obtained.

```
CSPEED speed
```

Speed – to set a fast clock rate; between 1 and 12 times normal time. Required.

To set the Default Direction for MOPS *[Supervisor access only]*

This sets the default direction for MOPS:

```
CHPARM DIRECTION;default[E/W/S/N/U/D]
```

Default – East, West, North, South, Up or Down. Required

To set the Passenger Car Class *[Supervisor access only]*

This allows MOPS to identify which Car Class is associated with Passenger Cars:

```
CHPARM PASSENGER;^car class^
```

Car Class – Car Class code that will be used to identify Passenger Cars. Required

To list all the parameters for MOPS

This lists parameters (sizes of fields, owning railroad name, current date/time, etc) for review. Listed in parameter name order.

```
LIPARM

PARAM=====   DATA=========================================================
AREASIZE     3
CLASSIZE     3
CSPEED       4
DATETIME     24JUN1972 13:52:00 SAT
LOADSIZE     4
RRNAME       ATLANTIC AND PACIFIC
STATSIZE     3
STATUS       STOPPED
STAXSIZE     4
```

To print all parameters for MOPS

This lists parameters (sizes of fields, owning railroad name, current date/time, etc) for review.

```
PRPARM
```

```
MOPS/PRPARM                        PARAMETERS                     25JUN1972 14:22
CC                             LIST OF PARAMETERS                      Page:    1

     PARAM=====   DATA=========================================================

     AREASIZE     3
     CARMAINT     120
     CARTSIZE     4
     CARWORKS     55
     CUROSIZE     3
     DATETIME     25JUN1972 14:22:20 SUN
     FLASHID      27
     LOADSIZE     4
     PLAXSIZE     2
     PRINTS       26
     RAILSIZE     3
     RRNAME       ATLANTIC AND PACIFIC
     SCHDSIZE     2
     STATUS       STOPPED
     STAXSIZE     4
```

To Stop the Background Processing on MOPS *[Supervisor access only]*

This stops the MOPS Background processes. This will stop the generation of commodities at warehouses, and tracking of fuel consumption and loco and car maintenance. The clock for MOPS has effectively been stopped. The banner 'MOPS running in Maintenance Mode' will be displayed and all users (apart from Supervisors) will be ejected from MOPS. The Help prompt (XXSTOP ?) is not available on this command.

```
XXSTOP
```

How to exit MOPS
To finish a user session on MOPS, enter quit or exit. Note that this isn't available for the background tasks – if these are running then you need to enter XXSTOP as a user command from another session. The Help prompt (QUIT ?) is not available on this command.

```
QUIT or EXIT
```

Help: some online assistance

Where to find Help
This has brief details on how to get a list of the available commands, and how to enter data. The Help prompt (HELP ?) is not available on this command.

```
HELP
```

What MOPS is about
This contains brief details about the MOPS and Python versions, and the location of the database. The copyright summary is available here. The Help prompt (ABOUT ?) is not available on this command.

```
ABOUT
```

List of Available Commands
This option displays all the available MOPS commands with their brief name, in alphabetical order, in groups of about 20 commands. The Help prompt (ASSIST ?) is not available on this command. Hit enter until all commands have been displayed.

```
ASSIST
```

Flash Messages: letting everyone know what's going on

This command allows you to send a message to all MOPS users. They will be displayed the next time a user uses a command, or if a user signs on. They are automatically deleted after 24 MOPS hours.

```
FLASHX message
```

Message – information to be sent to all other users

Geography

Railroads: Identifying ownership of Stations and Rolling Stock

Add a Railroad [Supervisor access only]
Add a Railroad and Railroad name. Railroads are owners of Stations (via Areas), Locomotives and passenger and Freight Cars.

```
ADRAIL railroad;railroad name
```

Railroad – code size is determined by the setting of **railsize**. Required.
Railroad name – name of the Railroad. Required.

Change a Railroad's name *[Supervisor access only]*
Change a Railroad's name.

```
CHRAIL railroad;railroad name
```

Railroad – code of record to be amended. Required.
Railroad name – name of the Railroad. Required.

Delete a Railroad *[Supervisor access only]*
Delete a Railroad. A Railroad cannot be deleted if it is currently linked to an Area, or if it is the owner of Cars or Locomotives.

```
DXRAIL railroad
```

Railroad – code of record to be deleted. Required

List Railroads
List Railroad codes and names to the screen.

```
LIRAIL (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Railroad Code order
1 – sorted in Railroad Name order

```
RAI NAME==========================
APR ATLANTIC AND PACIFIC RAILROAD
PRR THE PITTSFIELD RAILROAD
```

Print Railroads
List Railroads to a formatted report, ready for printing. Also reports the number of Areas linked to that Railroad, and the number of locos and cars owned by the Railroad.

```
PRRAIL (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Railroad Code order
1 – sorted in Railroad Name order

```
MOPS/PRRAIL                    ATLANTIC AND PACIFIC              24JUN1972 13:54
CC                   LIST OF RAILROADS SORTED BY RAILROAD CODE         Page:    1

            RAI NAME========================== AREAS= LOCOS= CARS==

            APR ATLANTIC AND PACIFIC RAILROAD       3      0      0
            PRR THE PITTSFIELD RAILROAD             1      0      0
```

Areas: Grouping Stations together

Add an Area *[Supervisor access only]*
Adds an Area code and name to the system. The Area must be allocated to a Railroad. An Area is a geographical grouping, normally for management purposes, that contains a number of Stations.

```
ADAREA area;area name;^railroad^
```

Area – code size is determined by the setting of **areasize**. Required.
Area name – describes the Area. Required.
Railroad – code of Railroad to which the Area belongs. Required

Change an Area's Name or Railroad *[Supervisor access only]*

Allows amendment of an Area Name, or the Railroad to which the Area is allocated.

```
CHAREA area;(area name);(^railroad^)
```

Area – code of record to be amended. Required.
Area name – name of the Area being changed. Optional.
Railroad – code of Railroad to which the Area belongs. Optional

Delete an Area *[Supervisor access only]*

Deletes an existing Area. If any Station is allocated to that Area, then that Area cannot be deleted.

```
DXAREA area
```

Area – code of Area record to be deleted. Required

List Areas

List Areas to the screen, and can be filtered to show Areas for a single railroad only. The listing shows the Railroad that each Area is linked to.

```
LIAREA (sort[0/1/2]);(^railroad^)
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
- 0 – sorted in Railroad Code, then Area Code order
- 1 – sorted in Area Code order
- 2 – sorted in Area Name order

Railroad – filter by Railroad: include Areas for that Railroad only. Optional.

```
ARE AREA NAME==============================   RAI RAILROAD NAME=================
SCH SCHODACK                                  APR ATLANTIC AND PACIFIC RAILROAD
SPR SPRINGFIELD                               APR ATLANTIC AND PACIFIC RAILROAD
PTF PITTSFIELD                                PRR THE PITTSFIELD RAILROAD
```

Print Areas

Print Areas to a report, and can be filtered to show Areas for a single railroad only. The report shows the Railroad to which each Area belongs.

```
PRAREA (sort[0/1/2]);(^railroad^)
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
- 0 – sorted in Railroad Code, then Area Code order
- 1 – sorted in Area Code order
- 2 – sorted in Area Name order

Railroad – filter by Railroad: include Areas for that Railroad only

```
MOPS/PRAREA                    ATLANTIC AND PACIFIC               24JUN1972 13:54
CC                        AREAS SORTED BY RAILROAD, AREA               Page:    1

ARE AREA NAME==============================   RAI RAILROAD NAME=================

SCH SCHODACK                                  APR ATLANTIC AND PACIFIC RAILROAD
SPR SPRINGFIELD                               APR ATLANTIC AND PACIFIC RAILROAD
STK STOCKBRIDGE                               APR ATLANTIC AND PACIFIC RAILROAD
PTF PITTSFIELD                                PRR THE PITTSFIELD RAILROAD
```

Station Types: Identifying What the Station Is

Add a Station Type [Supervisor access only]

Allows a Station Type and associated description to be added. The Station Type is used to define the nature of a Station, but has no operational significance.

ADSTAT station type;description

Station type – code size is determined by the setting of **statsize**. Required.
Description – description of the Station Type. Up to 30 characters long. Required.

Change a Station Type's Description [Supervisor access only]

Change the Description of a Station Type.

CHSTAT station type;description

Station type – code of record to be amended. Required.
Description – amend the description of the Station Type. Required.

Delete a Station Type [Supervisor access only]

Delete a Station Type from the system. If the Station Type is associated with a Station, then the deletion will not be allowed.

DXSTAT station type

Station type – code of record to be deleted. Required

List Station Types

List Station Types and their associated description to the screen.

LISTAT (sort[0/1])

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Station Type Code order
1 – sorted in Station Type Description order

```
TYP DESCRIPTION===================
I/C INTERCHANGE
JCN JUNCTION
STA STAGING
STN STATION
```

Print Station Types

List Station Types and their associated description. Also shows the number of Stations that are associated with that Station Type.

PRSTAT (sort[0/1])

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Station Type Code order
1 – sorted in Station Type Description order

```
MOPS/PRSTAT                    ATLANTIC AND PACIFIC              24JUN1972 13:54
CC                     LIST OF STATION TYPES SORTED BY CODE           Page:    1

                   TYP DESCRIPTION=================== STTNS=

                   I/C INTERCHANGE                         1
                   JCN JUNCTION                            4
                   STA STAGING                             9
                   STN STATION                             8
```

Stations: central to the operation of MOPS

Add a New Station *[Supervisor access only]*

Stations are 'Reporting Places' for MOPS, and indicate where Cars and Locomotives are located. They are used as reporting points for the progress of Trains. Cars and Locos are allocated to Stations as their Home Station. A Station has an abbreviated name and a long name. The Station is linked to an Area, and also a Type of Station. Stations can also act as aliases for other stations: for example, set a Staging or Fiddle Yard as a Station, then create additional 'real-world' Stations which are aliases for the Staging/Fiddle Yard.

```
ADSTAX station;short name;long name;^area^;^station type^;(^alias^)
```

Station - must have the same number of characters as set by staxsize. Required.
Short name - must be 8 characters or fewer. Required.
Long. name - must be 30 characters or fewer. required.
Area – area code as previously set up on MOPS. Required.
Station Type - Station Type Code, and must already exist on the system. Required.
Alias – Station Alias (must be an existing Station Code). Optional

Change a Station's details *[Supervisor access only]*

Change a Station's abbreviated name or long name, or link the station to a different area. The Type of station can also be changed. Stations can be linked to a new alias (to remove an existing alias, enter REMOVE as the station alias code.

```
CHSTAX station;(short name);(long name);(^area^);(^station type^);(^alias^)
```

Station – station code that is to be amended. Required.
Short name - must be 8 characters or fewer. Optional.
Long. name - must be 30 characters or fewer. Optional.
Area – area code as previously set up on MOPS. Optional.
Station Type - Station Type Code, and must already exist on the system. Optional.
Alias – Station Alias – Station code of station of which this Station is an alias. Optional

Delete a Station *[Supervisor access only]*

Allows a Station to be deleted. A Station cannot be deleted if places still exist that are linked to that station, if the Station is acting as an alias, or if there are any locomotives or cars that are linked (in any way) to that Station, or if the Station is part of a route. If a station is deleted, then any instructions linked to that station are also deleted.

```
DXSTAX station
```

station – code of Station to be removed from MOPS. Required.

List Stations

Lists basic station details to the screen, including Area, Station Type and Alias (if applicable).

```
LISTAX (sort[0/1/2/3]);(^area^)
```

Option – sort order for report. Optional
- 0 – sorted in Area Code / Station Code order
- 1 – Station Code order
- 2 – Short Name order
- 3 – Long Name order

Area – filter selected Stations to the input Area. Optional

```
ARE STAT SHORT NA LONG NAME===================== TYP ALIA
PTF 6000 FORSTJCN FORREST JUNCTION              JCN
PTF 6500 VALLEYDP VALLEY DEPOT                  STN
SCH 7000 SCHODACK SCHODACK                      STA
SCH 7100 ALBANYXX ALBANY                        STA 7000
```

Print Stations

Prints Stations to a file for printing on the default printer. Details include Area, Station Type and Alias.

`PRSTAX (sort[0/1/2/3]);(^area^)`

Option – sort order for report. Optional

0 – sorted in Area Code / Station Code order
1 – Station Code order
2 – Short Name order
3 – Long Name order

Area – filter selected Stations to the input Area. Optional

```
MOPS/PRSTAX                    ATLANTIC AND PACIFIC               24JUN1972 13:54
CC                        LIST STATIONS BY AREA, STATION               Page:    1

           ARE STAT SHORT NA LONG NAME===================== TYP ALIA

           PTF 6000 FORSTJCN FORREST JUNCTION               JCN
           PTF 6100 VALLEYYD VALLEY YARD                    STN
           PTF 6200 PITTSFLD PITTSFIELD                     STN
           PTF 6500 VALLEYDP VALLEY DEPOT                   STN
           SCH 7000 SCHODACK SCHODACK                       STA
```

Places: where stuff happens

Add a Place *[Supervisor access only]*

Add a Place to a Station, to define where railroad actions can take place. Placing a Car or Locomotive at specific places will cause actions to happen to that Car or Locomotive: locomotive refuelling (by type); loco and car maintenance; car cleaning, or loading / unloading of commodities. These are defined by the Type of Place. Place codes are unique within a Station; the same place codes can be used across multiple stations (for example, a diesel refuelling point can be defined with a place code of '55' across all Stations). A track length can be entered, although MOPS carries out no validation against what is standing on that track. A unique Place id is generated when the place is created.

`ADPLAX ^station^;place;name;place type[C/D/L/M/O/S/X];(track_length)`

Station – Station code that the place is attached to. Required.
Place – reference; only needs to be unique within a Station. Required.
Name – name of place (yard name, customer name, shed name). Required
Place type – type of activity undertaken at that place – Required

C – Clean Cars
D – Diesel refuelling
L – Locomotive maintenance
M – Car Maintenance
O – Other Refuelling location type
S – Steam Refuelling (coal, water)
X – No facilities

Track length – length of track at that place. Optional, defaults to zero

Change Details for a Place *[Supervisor access only]*

Amend details for a Place, changing the name or track length only. Once a Place has been established as a Type, that Type cannot be changed – to change the type, the Place has to be deleted then re-entered.

`CHPLAX ^station^;place;(name);(track_length)`

Station – Station code that the place is attached to. Required.
Place – reference; only needs to be unique within a Station. Required.
Name – name of place (yard name, customer name, shed name) Optional
Track length – length of track at that place. Optional, defaults to zero

Delete a Place *[Supervisor access only]*
Delete a Place from MOPS. The unique Id of the Place must be used to delete the Place, and it must not be linked to Warehouses or have any Locomotives or Cars at that Place.

```
DXPLAX id
```

id – unique id of record to be deleted. Required

List Places
List Places on MOPS to the screen. Places may be sorted as required, and can optionally be filtered by a specific Station. This listing includes industries, showing loading/unloading codes.

```
LIPLAX (sort[0/1/2]);(^station^)
```

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Station/Place Code order
1 – sorted in Place/Station code order
2 – sorted in Place Name order

Station – limit reported places to that station. Optional.

```
===ID STAT PL NAME========================== LNGTH INDUSTRY== LOAD UNLO
   10 1050 75 SPRINGFIELD YARD ARRIVALS       4000
   11 1050 76 SPRINGFIELD YARD HOLDING       12500
   12 1050 77 SPRINGFIELD YARD DEPARTS        4500
   13 1050 78 SPRINGFIELD YARD RIP TRCK       1540
    3 1100 85 RUSSEL ROAD DIESEL DEPOT         500 MAINT LOCO
```

Print Places
Prints Places on MOPS to a report. Places may be sorted as required, and can optionally be filtered by a specific Station. This listing includes industries, showing loading/unloading codes.

```
PRPLAX (sort[0/1/2]);(^station^)
```

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Station/Place Code order
1 – sorted in Place/Station code order
2 – sorted in Place Name order

Station – limit reported places to that station. Optional.

```
MOPS/PRSTAX                   ATLANTIC AND PACIFIC              24JUN1972 13:54
CC                     LIST OF PLACES SORTED BY STATION, CODE           Page:    1

    ===ID STAT PL NAME========================== LNGTH USAGE===== LOAD UNLO

       10 1050 75 SPRINGFIELD YARD ARRIVALS       4000
       12 1050 77 SPRINGFIELD YARD DEPARTS        4500
        3 1100 85 RUSSEL ROAD DIESEL DEPOT         500 MAINT LOCO
       22 2100 01 MOUNTAIN HIGH ALUMINIUM          450 MOUNTAINAL RAMP
        8 4100 92 STOCKBRIDGE CAR SHOP               0 MAINT CARS
       23 5000 01 FELHAM CANNING                   130 FELHAMCAN  RAMP RAMP
```

List Geography

Report a list of railroads; within each railroad reports Areas; within each Area reports Stations, and within each Station reports Places. The Place ids are shown in brackets, with a '+' sign.

LIGEOG

```
APR ATLANTIC AND PACIFIC RAILROAD
    SCH SCHODACK
        7000 SCHODACK
        7100 ALBANY
        7200 SYRACUSE
        7300 ROCHESTER
    SPR SPRINGFIELD
        1000 SPRINGFIELD
        1050 SPRINGFIELD YARD
            75 SPRINGFIELD YARD ARRIVALS (+10)
            76 SPRINGFIELD YARD HOLDING (+11)
            77 SPRINGFIELD YARD DEPARTS (+12)
            78 SPRINGFIELD YARD RIP TRCK (+13)
        1100 HARTFORD
```

Print Geography

Report a list of railroads; within each railroad reports Areas; within each Area reports Stations, and within each Station reports Places.

PRGEOG

```
MOPS/PRGEOG                      ATLANTIC AND PACIFIC                 24JUN1972 13:54
CC                           GEOGRAPHICAL LIST OF PLACES                   Page:    1

                           RAILROAD / AREA / STATION / PLACE

                           APR ATLANTIC AND PACIFIC RAILROAD
                               SCH SCHODACK
                                   7000 SCHODACK
                                   7100 ALBANY
                                   7200 SYRACUSE
                                   7300 ROCHESTER
                               SPR SPRINGFIELD
                                   1000 SPRINGFIELD
                                   1050 SPRINGFIELD YARD
                                       75 SPRINGFIELD YARD ARRIVALS (+10)
                                       76 SPRINGFIELD YARD HOLDING (+11)
                                       77 SPRINGFIELD YARD DEPARTS (+12)
                                       78 SPRINGFIELD YARD RIP TRCK (+13)
                                   1100 HARTFORD
                                       85 RUSSEL ROAD DIESEL DEPOT (+3)
                                   1200 BOSTON
```

Industry

Loading Codes: moving commodities in and out of cars

Add a Loading Code [Supervisor access only]

Loading codes are used to define how Commodities are loaded into Cars, what facilities Places have to Load and Unload Commodities, and wether Cars can carry a given Commodity. Add a Loading Code (with a description) and indicate whether the loading code can be used for loading only, unloading only, or for both loading and unloading.

```
ADLOAD load;description;loading[Y/N];unloading[Y/N]
```

Load – code size is determined by the setting of **loadsize**. Required.
Description – description of the Loading Code. Required.
Use for loading – y/n whether the code can be used for loading commodities. Required
Use for unloading – y/n whether the code can be used for unloading commodities. Required

Change a Loading Code's details [Supervisor access only]

Allows a Loading Code's description, and whether used for loading and/or unloading, to be modified.

```
CHLOAD load;(description);(loading[Y/N]);(unloading[Y/N])
```

Load – code of record to be amended. Required.
Description – amend the description of the Loading Code. Optional.
Use for loading – y/n whether the code can be used for loading commodities. Optional
Use for unloading – y/n whether the code can be used for unloading commodities. Optional

Delete a Loading Code [Supervisor access only]

Delete a Loading Code from the system. A Loading Code cannot be deleted if it is in use at a Place, by a Commodity or by a Car Type.

```
DXLOAD load
```

load – code of record to be deleted. Required

List Loading Codes

List Loading Codes available on MOPS. The listing also indicates how many instances of the loading code is used by Places, Commodities and Cars.

```
LILOAD (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Loading Code order
1 – sorted in Loading Description order

```
LOAD DESCRIPTION===================   LOAD UNLOAD PLACE CMDTY =CARS
APRN CONCRETE APRON (TRACK LEVEL)        Y      Y     0     0     0
AUTO AUTOMOBILE LOADING                  Y      Y     0     0     0
BINS GRAVITY DISCHARGE BINS              N      Y     0     0     0
CONT CONTAINER LIFTING                   Y      Y     0     0     0
HOPP GRAVITY HOPPER FEED                 Y      N     0     0     0
HOSE HOSE FILL (TOP LOADING)             Y      N     3     4     0
HOSX HOSE EMPTY (SIDE UNLOADING)         N      Y     2     0     0
LIFT CRANE LIFT (GENERAL USE)            Y      Y     0     0     0
```

Print Loading Codes

Print a list of Loading Codes to a report for printing.

> **PRLOAD (sort[0/1])**
>
> Sort –sort order of data. Optional; defaults to 0 if not entered.
>
> 0 – sorted in Loading Code order
>
> 1 – sorted in Loading Description order

```
MOPS/PRLOAD                   ATLANTIC AND PACIFIC                  24JUN1972 13:54
CC                         LIST OF LOADING TYPE CODES                    Page:    1

               LOAD DESCRIPTION===================   LOAD UNLOAD

               APRN CONCRETE APRON (TRACK LEVEL)        Y      Y
               AUTO AUTOMOBILE LOADING                  Y      Y
               BINS GRAVITY DISCHARGE BINS              N      Y
               CONT CONTAINER LIFTING                   Y      Y
               HOPP GRAVITY HOPPER FEED                 Y      N
               HOSE HOSE FILL (TOP LOADING)             Y      N
               HOSX HOSE EMPTY (SIDE UNLOADING)         N      Y
               LIFT CRANE LIFT (GENERAL USE)            Y      Y
```

Commodities: the goods that move

Add a Commodity *[Supervisor access only]*

Adds a Commodity with a description. Commodities are the goods that are moved on the railroad. These records define how the commodity is loaded into cars using a Loading Code. A Commodity is also associated with loading and unloading rates, and whether the commodity requires clean cars. Loading and Unloading rates are hourly rates and should include 'loiter' time at the customer. If a commodity requires a Clean Car and a Clean Car is not provided, then the Commodity will not be loaded.

> **ADCOMM commodity;description;^load^;loading rate; unloading rate; clean car[Y/N]**
>
> Commodity – code size is determined by the setting of **commsize**. Required.
>
> Description – description of the Commodity. Required.
>
> Load – loading code to be used for putting the commodity in cars. Required
>
> Loading rate – rate at which commodity can be loaded in tons/hours. Required.
>
> Unloading rate – rate at which commodity can be unloaded in tons/hours. Required.
>
> Clean car – Y/N whether a clean car is required for loading commodity. Required

Change a Commodity's details *[Supervisor access only]*

Changes a commodity's name, loading code, loading or unloading rate, and whether a clean car is required. The Loading and Unloading Rate will be per hour.

> **CHCOMM commodity;(description);(^load^);(loading rate); (unloading rate);(clean car[Y/N])'**
>
> Commodity – code of record to be amended. Required.
>
> Description – amend the description of the Loading Code. Optional.
>
> Load – loading code to be used for putting the commodity in cars. Optional
>
> Loading rate – rate at which commodity can be loaded in tons/hours. Optional.
>
> Unloadding rate – rate at which commodity can be unloaded in tons/hours. Optional.
>
> Clean car – Y/N whether a clean car is required for loading commodity. Optional

Delete a Commodity *[Supervisor access only]*

Deletes a commodity from the system. Deletion is not allowed if the Commodity is being transported (on a Car) or is in use by a warehouse.

```
DXCOMM  commodity
```

Commodity – code of record to be deleted. Required

List Commodities

Lists Commodities to the screen, twenty records at a time

```
LICOMM (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Commodity Code order

1 – sorted in Commodity Description order

```
COM DESCRIPTION==================  LOAD    LOAD  UNLOAD CLEAN?
100 BAGGED GRAIN ON PALLETS        RAMP       5       4      Y
200 PALLETED MACHINE TOOLS         RAMP      45      45      N
220 PALLETED AUTO PARTS            RAMP      45      45      N
300 WOOD PULP                      HOSE      42      33      N
315 INK PRODUCTS                   HOSE      73      73      N
320 PAPER (ROLLS)                  RAMP      55      55      N
```

Print Commodities

Prints Commodities to a report for printing on the default printer.

```
PRCOMM (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Commodity Code order

1 – sorted in Commodity Description order

```
MOPS/PRCOMM                  ATLANTIC AND PACIFIC              24JUN1972 13:54
CC                           LIST OF COMMODITIES                    Page:    1

         COM DESCRIPTION==================  LOAD    LOAD  UNLOAD CLEAN?

         100 BAGGED GRAIN ON PALLETS        RAMP       5       4      Y
         200 PALLETED MACHINE TOOLS         RAMP      45      45      N
         220 PALLETED AUTO PARTS            RAMP      45      45      N
         300 WOOD PULP                      HOSE      42      33      N
         315 INK PRODUCTS                   HOSE      73      73      N
         320 PAPER (ROLLS)                  RAMP      55      55      N
```

Industry: creating commodities to move

Add an Industry *[Supervisor access only]*

Add the details of an Industry: an Industry is a specific type of Place at which goods can be loaded or unloaded. The station and place code, a place name and a unique code for the industry are required. Also provide track length, and loading and unloading codes (if not provided, then loading and/or unloading as appropriate cannot take place at that location).

```
ADINDY station;place;place name;length;industry;(^load^);(^load^)
```

Station – code of station at which industry is located. Required.
Place – place code (within Station) of the industry. Required.
Place name – name of place at which the industry is located. Required.
Length – length of track for cars. Required.
Industry – code name for industry; must be unique and 10 characters or fewer. Required
Load (1st) – code for loading mechanism at that place. Optional (if not provided, no loading allowed)
Load (2nd) – code for unloading at that place. Optional (if not provided, no unloading allowed)

Change an Industry's Name or other details *[Supervisor access only]*

Allows an Industry's details to be modified: Station/Place cannot be modified (to 'move' an Industry, delete and re-input), but allows changes to place name, track length, loading and unloading mechanisms and the industry code.

```
CHINDY station;place;(place name);(track length);(^industry^);(^load^)
;(load)
```

Station – code of station at which industry is located. Required.
Place – place code (within Station) of the industry. Required.
Place name – name of place at which the industry is located. Optional.
Length – length of track for cars. Optional.
Industry – code name for industry; must be unique on file and 10 characters or fewer. Optional
Load (1st) – code for loading mechanism at that place. Optional (if not provided, no loading allowed)
Load (2nd) – code for unloading at that place. Optional (if not provided, no unloading allowed)

Delete an Industry

[Supervisor access only]

Delete an Industry from MOPS. It can't be deleted if it's referenced by another Industry (as a Destination) or if there is a Warehouse attached to it.

```
DXINDY place id
```

Place id – unique place id of industry to be deleted. Required

There are no specific industry listing commands: use the Place listing commands LIPLAX and PRPLAX instead; these contain Industry (as well as other) information

Customer Routing: How the Customer wants the Goods to get there

Add a Customer Routing Description *[Supervisor access only]*

The Customer Routing Description consists of a code and a description. The Customer Routing is the intended route of the Car (eg it becomes Waybill information). There are no links between the information held on this record and any other Station, Routing or Schedule details: this is informational data only.

```
ADCURO customer routing;description
```

Customer routing – code for routing, size depends on setting of **curosize**. Required
Description – free-form field with information on routing. Required

Amend a Customer Routing Description *[Supervisor access only]*

Amends the Customer Routing Description.

```
CHCURO customer routing;description
```

Customer routing – routing code that requires to be amended. Required
Description – free-form field with information on routing. Required

Delete a Customer Routing Description *[Supervisor access only]*

Removes a Customer Routing Description from MOPS.

```
DXCURO customer routing
```

Customer routing – customer routing code to be removed from MOPS

List Customer Routing Descriptions

List costumer routing codes to the screen. Sortable.

```
LICURO (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Code order
1 – sorted in Description order

```
COD ROUTING==========================================
001 STCKHT-STCKBR-SPRFLD-PRVDNC
002 PTSFLD-PTFDIC-STCKBR-SPRFLD-PRVDNC
003 ALBANY-PTFDIC-VALLYD-PTSFLD
004 SYRCSE-PTFDIC-VALLYD-PTSFLD
```

Print Customer Routing Descriptions

Prints Customer Routing codes and descriptions to a report. The report is sortable.

```
PRCURO (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Code order
1 – sorted in Description order

```
MOPS/PRCURO                   ATLANTIC AND PACIFIC              24JUN1972 13:54
CC                         LIST OF CUSTOMER ROUTINGS                 Page:    1

             COD ROUTING==========================================

             001 STCKHT-STCKBR-SPRFLD-PRVDNC
             002 PTSFLD-PTFDIC-STCKBR-SPRFLD-PRVDNC
             003 ALBANY-PTFDIC-VALLYD-PTSFLD
             004 SYRCSE-PTFDIC-VALLYD-PTSFLD
```

Warehouses: Where Commodities are prepared for dispatch

Add a Warehouse [Supervisor access only]

A Warehouse is located at an Industry (an Industry can support a number of warehouses). A Warehouse creates and stores Commodities, issues requests for Empty Cars/Waybills, and loads commodities. A Warehouse has a production rate for the Commodity, and also has a threshold at which point the Warehouse will request empty car(s) of a specific Class to go to a specific destination (another Industry). The loading code for the commodity must agree with the loading code for the Industry. If the commodity total at the warehouse reaches maximum storage then production stops at that warehouse until it is reduced back below the maximum threshold. The Customer Routing code for the loaded cars is also required. Warehouses with the same Industry code, Commodity and Destination are not allowed. A Warehouse is created with a unique id.

```
ADWARE ^industry^;^commodity^;^destination^;prod rate;threshold quantity;
'cars to order;^class of car^;max storage;^customer routing^
```

Industry – industry code of the place at which the warehouse is located. Required
Commodity – goods that are being produced at that warehouse. Required
Destination – industry code of the place to which the goods are being sent. Required
Prod rate – hourly production rate of the goods. Required
Threshold quantity – trigger amount for ordering empty cars. Required
Cars to order – number of cars requested when threshold quantity reached. Required
Class of cars – class of car requested when threshold quantity reached. Required
Max storage – maximum amount of goods that will be held at the warehouse. Required
Routing – customer routing code for adding to waybill. Required

Change Details for a Warehouse [Supervisor access only]

Amend the details for a warehouse, including details of commodity, destination, production rates and threshold, details of cars to be ordered and customer routing code. Note that if cars are despatched and details are subsequently changed, this will not affect cars en route.

```
CHWARE warehouse id;(^industry^);(^commodity^);(^destination^);(prod rate);
(threshold quantity);(cars to order);(^class of cars^);(max
storage);(^customer routing^)
```

Warehouse id – unique id of warehouse being amended. . Required
Industry – industry code of the place at which the warehouse is located. Optional
Commodity – goods that are being produced at that warehouse. Optional
Destination – industry code of the place to which the goods are being sent. Optional
Prod rate – hourly production rate of the goods. Optional
Threshold quantity – trigger amount for ordering empty cars. Optional
Cars to order – number of cars requested when threshold quantity reached. Optional
Class of cars – class of car requested when threshold quantity reached. Optional
Max storage – maximum amount of goods that will be held at the warehouse. Optional
Routing – customer routing code for adding to waybill. Optional

Delete a Warehouse [Supervisor access only]

Removes a Warehouse from MOPS. Note that deleting a warehouse will not affect any cars en route to the industry at which this Warehouse is attached or to another (destination) industry. The unique idd of the warehouse is required to delete it.

```
DXWARE id
```

id – Unique id of warehouse to be deleted. Required.

Change Production at a Warehouse[Supervisor access only]

Allows quick change of the Production Rate (eg seasonal changes; holiday effects).

```
CPWARE warehouse id;prod rate
```

Warehouse id – unique id of warehouse being amended. . Required
Prod rate – hourly production rate of the goods. Required

List Warehouses

List warehouse details to the screen, including warehouse id and production details. Sortable by Unique id or by station/place.

```
LIWARE (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Warehouse id order
1 – sorted in Station/Place order

```
15:28 LIST WAREHOUSES BY BF ON 25JUN1972
STAT PL ================== INDUSTRY== WAREH COMM PRODN MAX== ORDER CARS= CLA
7000 01 ALBANY DERIVATIVES ALBANYRAMP     1 500      2   600   120     1 BOX
7000 02 ALBANY DERIVATIVES ALBANYHOSE     2 315      4   400   200     2 TNK
7100 01 SYRACUSE CANNED FO SYRACUSECF     3 550      1   300   100     1 BOX
7100 02 ROCHESTER ALUMINIU ROCHESTERA     4 600      2   300   150     2 GON
```

List Warehouse Details

List warehouse production and other details to the screen, including warehouse id and production details. Sortable by Unique id or by station/place.

```
LDWARE (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Warehouse id order
1 – sorted in Station/Place order

```
STAT PL INDUSTRY== WAREH CAPY% ORDER ONWAY EMPTY LOADS TRACK USED. ROU
7000 01 ALBANYRAMP     1     1     0     0     0     0   115     0 C06
7000 02 ALBANYHOSE     2     3     0     0     0     0   120     0 C03
7100 01 SYRACUSECF     3     1     0     0     0     0   335     0 C03
7100 02 ROCHESTERA     4     2     0     0     0     0   420     0 C01
7100 02 ROCHESTERA     5     1     0     0     0     0   420     0 C02
6200 01 PITTSFDCAN     6     3     0     0     0     0   420     0 C05
6200 02 PITTSFDCHM     7     1     0     0     0     0    95     0 C05
6200 02 PITTSFDCHM     8     1     0     0     0     0    95     0 C05
4250 01 STOCKBCOAL     9     0     0     0     0     0  2500     0 C06
4250 02 STOCKBPAPR    10     3     0     0     0     0   100     0 C04
4250 02 STOCKBPAPR    11     1     0     0     0     0   100     0 C04
```

Additional notes: CAPY% is the amount of goods in storage compared to the capacity of the warehouse. ORDER is the number of cars requested, ONWAY the number en route. EMPTY and LOADS indicate the loaded/empty cars at that location. The TRACK is the track length, and USED. Is the amount of track length used by cars. ROU is the customer routing code for these shipments.

List Warehouse Information

List full warehouse details showing station, place, route and production information. These details are available for a single station only and show all warehouses at that station.

```
LSWARE ^station^
```

Station – station code for which the warehouse list is required. Required.

```
STATION:4250 STATION NAME:STOCKBRIDGE HEIGHTS
-------------------------------------------------------------
PLACE:01 NAME:STOCKBRIDGE HEIGHTS COAL  TRACK LENGTH: 2500   LOADING:COAL
DEST:2060 PLACE:01 AVEBURY POWER STATION                     UNLOADING:COAX
WAREHOUSE:    9 STOCKBCOAL SHIPS TO:AVEBURYPWR SHIPPING:800 COAL
CAR CLASS:HOP CUSTOMER ROUTE:C06 S/BR HEIGHTS-S/BRIDGE JCN/RSL RD JCN-AVEBURY POWER
PROD RATE:   15 ORDER AT: 1200 STORED:  150 MAX STORE:33000 CARS ORDERED:    0

PLACE:02 NAME:STOCKBRIDGE PAPER MILL    TRACK LENGTH:  100   LOADING:RAMP
DEST:1100 PLACE:01 HARTFORD PRINTING                         UNLOADING:RAMP
WAREHOUSE:   10 STOCKBPAPR SHIPS TO:HRTFRDRAMP SHIPPING:320 PAPER (ROLLS)
CAR CLASS:BOX CUSTOMER ROUTE:C04 S/BRIDGE HGHTS-S/BRIDGE/JCN-SPRINGFIELD
PROD RATE:    3 ORDER AT:  280 STORED:   30 MAX STORE:  300 CARS ORDERED:    0
```

Print Warehouses

Print warehouses details. Can be sorted by Warehouse id or by Station/Place.

```
PRWARE (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Warehouse id order

1 – sorted in Station/Place order

```
MOPS/PRWARE                  ATLANTIC AND PACIFIC              24JUN1972 13:54
CC                            LIST OF WAREHOUSES                    Page:    1

STAT PL ====================== INDUSTRY== WAREH COM PRODN MAX== ORDER CARS= CLA

2100 01 MOUNTAIN HIGH ALUMINIUM MOUNTAINAL     1 600     4  2200   220      3 BOX
4250 01 STOCKBRIDGE HEIGHTS WOO STOCKBWOOD     2 300     2   500   100      1 BOX
6200 01 PITTSFIELD CHEMICALS (R PITTSFLD_R     3 430     1   250   140      1 BOX
7100 01 ALBANY CHEMICAL DERIVAT ALBANYCHEM     4 420     3   700   250      2 BOX
```

Empty Orders: Getting Empty Cars to the Customer

List Unfulfilled Empty Orders

Lists empty orders where a specific car or cars has/have not yet been assigned to the order. The list is printed in time order, oldest outstanding order listed first.

```
LEMPTY
```

```
ORDER INDUSTRY== STATION= COM CARS= CLN DESTINATN= DEST STN REQUESTED======
    1 ALLENHAYES SYRACUSE 420     1   N PITTSFLD_H PITTSFLD 24JUN1972 14:38
    2 EMOUNTAIN  ROCHESTR 500     1   N FELHAMCAN  FELHAMJN 24JUN1972 14:44
    3 PROVIDRAMP PROVDNCE 320     1   N FELHAMCAN  FELHAMJN 24JUN1972 14:44
    4 STOCKBWOOD STKBRGHT 300     1   N PROVIDHOSE PROVDNCE 24JUN1972 14:44
```

Print Unfulfilled Empty Orders

Lists empty orders where a specific car has not yet been assigned to the order. The list is printed in time order, oldest outstanding order listed first.

PEMPTY

```
MOPS/PEMPTY                    ATLANTIC AND PACIFIC              24JUN1972 14:51
CC                         LIST DEMANDS FOR EMPTY CARS                Page:    1

  ORDER INDUSTRY== STATION= COM CARS= CLN DESTINATN= DEST STN REQUESTED======

      1 ALLENHAYES SYRACUSE 420     1   N PITTSFLD_H PITTSFLD 24JUN1972 14:38
      2 EMOUNTAIN  ROCHESTR 500     1   N FELHAMCAN  FELHAMJN 24JUN1972 14:44
      3 PROVIDRAMP PROVDNCE 320     1   N FELHAMCAN  FELHAMJN 24JUN1972 14:44
      4 STOCKBWOOD STKBRGHT 300     1   N PROVIDHOSE PROVDNCE 24JUN1972 14:44
      5 MOUNTAINAL RSSLROAD 600     3   N FELHAMCAN  FELHAMJN 24JUN1972 14:47
```

Detail Empty Car Request

Lists Details about an assignment. If cars have been attached to an order then a list of these cars, with their current location, block and status (loaded, empty), will also be shown.

DEMPTY (^order^)

Order – order id for which information is required

```
ID:2  ORIGIN:STOCKBCOAL STKBRGHT  DEST:AVEBURYPWR AVEBURYP
COMMODITY: 800 COAL  LOAD: COAL  UNLOAD: COAX  CLEAN CARS? N
REQUIRES: 40 HOP  TRACK:2500 OCCUPIED:0
ORDER ORIGINALLY RAISED 26JUN1972 10:00
```

List all Empty Orders and Waybills

This listing displays details of all empty car requests and waybills.

LORDER

```
ORDER TYPE STAT INDUSTRY== ORIG/PL DESTINATN= DEST/PL CLA CLN   O/S
    1 MTY  O/S  ALBANYHOSE 7000/02 HRTFRDHOSE 1100/02 TNK  N      2
    2 MTY  O/S  STOCKBCOAL 4250/01 AVEBURYPWR 2060/01 HOP  N     40
```

Locomotives

Locomotive Types: Identifying the nature of power on the railroad

Add a Locomotive Type [Supervisor access only]

Adds a new Type of Locomotive. Each locomotive added to MOPS has to be designated as a type, which includes information about haulage capacity, length and weight, fuel, maintenance and how the locomotive operates (eg independently, or as part of a multiple unit) for that type of locomotive.

```
ADLOCT loco type;name;power type[D/S/E/O];haulage;fuel capacity;fuel rate;
maint interval;works time;weight;length;oper mode[I/S/M/O]
```

Loco type – locomotive type code size is determined by the setting of **loctsize**. Required.
Name – description of the type of locomotive. Required.
Type of power – Nature of the power unit. Required:
  Electric, this is an electric locomotive
  Diesel, this is a Diesel powered locomotive
  Steam, this is a steam-powered locomotive
  Other, locomotive not covered by one of the categories, above.
Haulage – weight that a locomotive can pull. Required
Fuel capacity – amount of fuel a locomotive can carry. Required
Fuel rate – rate at which fuel is used when on trains. Required
Maintenance interval. Days between maintenance overhaul. Required.
Works time. During overhaul, the number of Hours that the unit is unavailable. Required.
Weight. Weight of locomotive (used when locomotive is unpowered and on a train). Required.
Length. Length of locomotive. Required.
Operating mode. Indicates how a locomotive operates: Required.
  I – Independent (can operate on its own)
  D – Dummy/Slave; provides power but must operate with an independent loco in tandem
  M – forms part of a multiple set (ie at least two of the same units together)
  O – Other (treated as Independent by MOPS)

Change a Locomotive Type's details [Supervisor access only]

Change details about Locomotive Types. Each locomotive added to MOPS has to be designated as a type, which includes information about haulage capacity, length and weight, fuel, maintenance and how the locomotive operates (eg independently, or as part of a multiple unit).

```
CHLOCT loco type;(name);(power type[D/S/E/O]);(haulage);(fuel capacity);
(fuel rate);(maint interval);(works time);(weight);(length);
(oper mode[I/S/M/O])
```

Loco type – locomotive type to be amended. Required.
Name – description of the type of locomotive. Optional.
Type of power – Nature of the power unit. Optional:
  Electric, this is an electric locomotive
  Diesel, this is a Diesel powered locomotive
  Steam, this is a steam-powered locomotive
  Other, locomotive not covered by one of the categories, above.
Haulage – weight that a locomotive can pull. Optional
Fuel capacity – amount of fuel a locomotive can carry. Optional
Fuel rate – rate at which fuel is used when on trains. Optional
Maintenance interval. Days between maintenance overhaul. Optional.
Works time. During overhaul, the number of Hours that the unit is unavailable. Optional.
Weight. Weight of locomotive (used when locomotive is unpowered and on a train). Optional.
Length. Length of locomotive. Required Optional
Operating mode. Indicates how a locomotive operates: Optional.
  I – Independent (can operate on its own)
  D – Dummy/Slave; provides power but must operate with an independent loco in tandem
  M – forms part of a multiple set (ie at least two of the same units together)
  O – Other (treated as Independent by MOPS)

Delete a Locomotive Type *[Supervisor access only]*

Allows a Locomotive Type to be deleted. There must be no locomotives on MOPS designated as that Locomotive Type.

`DXLOCT loco type`

Loco type – code of record to be deleted. Required

List Locomotive Types

Lists locomotive types and their details to the screen

`LILOCT (sort[0/1])`

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Loco Type Code order

1 – sorted in Loco Type Name order

```
TYPE= LOCOMOTIVE TYPE========= PWR HAULS F.CAP FRATE MAINT WORKS WEGHT LNGTH MOD
8-40B DASH 8-40B               DSL  4000  2300    20    90    45   280    59 IND
E6    EMD E6                   DSL  2000  1800    22    90    45   245    57 IND
E6B   EMD E6B SLAVE            DSL  2000  1800    22    90    40   242    57 DUM
```

Print Locomotive Types

Extracts and prints Locomotive Types to a print file, reading for printing to the default printer.

`PRLOCT (sort[0/1])`

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Loco Type Code order

1 – sorted in Loco Type Name order

```
MOPS/PRLOCT                    ATLANTIC AND PACIFIC               24JUN1972 13:54
CC                           LIST OF LOCOMOTIVE TYPES                  Page:    1

TYPE= LOCOMOTIVE TYPE========= PWR HAULS F.CAP FRATE MAINT WORKS WEGHT LNGTH MOD

8-40B DASH 8-40B               DSL  4000  2300    20    90    45   280    59 IND
E6    EMD E6                   DSL  2000  1800    22    90    45   245    57 IND
E6B   EMD E6B SLAVE            DSL  2000  1800    22    90    40   242    57 DUM
GP7   EMD GP-7                 DSL  2500  1600    18    90    45   145    56 IND
P42DC GE P42DC                 DSL  4200  2200    16   120    35   268    70 IND
```

Locomotives: the power on the road

Add a Locomotive *[Supervisor access only]*

Adds a new Locomotive. Each locomotive added to MOPS is designated as a Type, which includes information about haulage capacity, length and weight, fuel, maintenance and how the locomotive operates (eg independently, or as part of a multiple unit). The other information added for a locomotive are its owning Railroad and its designated Home Station. The current Station (ie where it is now) defaults to its Home Station when created. Other information is also defaulted (eg fuel is set to maximum value; time to maintenance defaults to maximum value), or is set by other commands as required.

`ADLOCO loco;^loco type^;^railroad^;^home station^`

Loco – locomotive running number; code size is determined by the setting of **locosize**. Required.
Type of locomotive – class of locomotive as defined by its type. Required.
Railroad – owning railroad. All rolling stock is owned by a railroad (at least nominally) Required.
Home Station – place a locomotive will return to when not in use. Required.

Change a Locomotive Fuel load *[Supervisor access only]*

Correcting Command. Allows fuel on board a locomotive to be changed manually. Normally a loco will be fuelled automatically at a refuelling point. This allows for the value to be set independently of normal refuelling.

```
FUELXX loco;fuel
```

Loco – running number of locomotive to be changed. Required
Fuel – amount of fuel on board locomotive (up to max value for type). Required.

Change Locomotive details *[Supervisor access only]*

Allows changes to the Locomotive Type, Owning Railroad or Home Station. Other commands deal with changes to other Locomotive information eg Maintenance state, Power state, current Station, etc.

```
CHLOCO loco;(^loco type^);(^railroad^);(^home station^)
```

Loco – running number of locomotive to be changed. Required
Type of locomotive – class of locomotive as defined by its type. Optional.
Railroad – owning railroad. All rolling stock is owned by a railroad (at least nominally) Optional.
Home Station – place a locomotive will return to when not in use. Optional.

Change a Locomotive's Power State

Allows the Power state of a locomotive to be changed. This indicates whether a locomotive is producing its own power or is, in effect, a 'dead weight' on a train.

```
POWERX loco;power state [P/U]
```

Loco – running number of locomotive to be changed. Required
Power – indicates whether a locomotive is producing power or is a dead weight. Required.
P – Powered, locomotive is capable of haulage
U – Unpowered, locomotive is not producing power (eg failed)

Change a Locomotive's Maintenance Information *[Supervisor access only]*

Correcting Command. Allows the Maintenance state of a locomotive to be changed. This is either the number of days until a locomotive is due maintenance, or the number of hours until it is back in traffic. Although two times of information can be entered, only the first valid time is accepted: that is, if a locomotive is in works, then only works time figure will be updated for the locomotive, if not in works, then only the time until maintenance will be accepted.

```
MAINTL loco;(time to maint);(works time)
```

Loco – running number of locomotive to be changed. Required
Time to maintenance - number of days until locomotive requires servicing OR
Works time – number of hours until locomotive is in traffic

Change a Locomotive's Current Station *[Supervisor access only]*

Correcting command. Allows the Current Station of a locomotive to be changed. Changing the Current Station of a locomotive in this manner will automatically reduce the fuel on board the locomotive by 25% (to represent the transit to the new station). If the locomotive is undergoing maintenance, then the change is not allowed.

```
LOCOAT loco;^station^
```

Loco – running number of locomotive to be changed. Required
Station – new Station location for the locomotive. Required

Spot a Locomotive

Allows a locomotive to be located at a Place. If the Place is a Maintenance Place (for Locomotives) or a Refuelling Point (for the type of Locomotive), then Maintenance or Refuelling will begin automatically. The place allocated will be the place within the loco's current station.

```
LOCOSP loco;^place^
```

Loco – running number of locomotive to be changed. Required
Place – Place code (within the Station) where the Locomotive is to be located. Required.

Delete a Locomotive *[Supervisor access only]*

Allows a Locomotive to be deleted. The Locomotive cannot be on a Train.

```
DXLOCO loco
```

Loco – running number of locomotive to be changed. Required

List Locomotive Data

Lists locomotives and their details to the screen. The data can be filtered by Locomotive Type, Railroad, Home Station or any combination of the three. This report gives 'static' details about the locomotives, such as their type, notes, owning railroad and home station.

```
LILOCO (sort[0/1/2]);(^loco type^);(^railroad^);(^station^)
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
- 0 – sorted in Locomotive running number order
- 1 – sorted in Locomotive Type, then Loco number order
- 2 – sorted by Railroad, then Locomotive Number

Loco type – filter by type of locomotive (as held on Locomotive Types). Optional
Railroad – filter by Owning Railroad. Optional
Home Station – filter by Home Station of locomotive. Optional

```
LOCO= TYPE== ENGINE   RR= HOME
06501 E6     DIESEL   APR 4100
38205 GP38/2 DIESEL   APR 4100
40319 DD40AX DIESEL   APR 4100
99001 GP38/2 DIESEL   APR 4100
E6001 E6     DIESEL   PRR 6200
E6900 E6B    DIESEL   PRR 6200
```

List Locomotive Status

Lists locomotive status details to the screen. The data can be filtered by Locomotive Type, Railroad, Home Station or any combination of the three. This report gives 'variable' details about the locomotives, such as remaining fuel, days to maintenance, time left in works (if in maintenance), power state (Powered, Unpowered, In Maintenance or in Refuelling), Current Station and Place.

```
LSLOCO (sort[0/1/2]);(^loco type^);(^railroad^);(^station^)
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
- 0 – sorted in Locomotive running number order
- 1 – sorted in Locomotive Type, then Loco number order
- 2 – sorted by Railroad, then Locomotive Number

Loco type – filter by type of locomotive (as held on Locomotive Types). Optional
Railroad – filter by Owning Railroad. Optional
Home Station – filter by Home Station f locomotive. Optional

```
LOCO= TYPE== FUEL= MAINT RPAIR PWR STAT PLACE======================== TRAIN=====
06501 E6      1800    90     0   P 4100                                0
38201 GP38/2  2600    90     0   P 4100                                0
40319 DD40AX  8230   120     0   P 4100                                0
E6001 E6      1800    90     0   P 6200                                0
```

Print Locomotive Data

Print version of List Locomotive Data. The data can be filtered by Locomotive Type, Railroad, Home Station or any combination of the three. This report gives 'static' details about the locomotives, such as their type, notes, owning railroad and home station.

`PRLOCO (sort[0/1/2]);(^loco type^);(^railroad^);(^station^)`

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Locomotive running number order
1 – sorted in Locomotive Type, then Loco number order
2 – sorted by Railroad, then Locomotive Number

Loco type – filter by type of locomotive (as held on Locomotive Types). optional
Railroad – filter by Owning Railroad. Optional
Home Station – filter by Home Station f locomotive. Optional

```
MOPS/PRLOCO                  ATLANTIC AND PACIFIC              24JUN1972 14:56
CC                        LOCOMOTIVES SORTED BY LOCO NUMBER         Page:    1

                         LOCO= TYPE== ENGINE   RR= HOME

                         06501 E6     DIESEL   APR 4100
                         38201 GP38/2 DIESEL   APR 4100
                         40319 DD40AX DIESEL   APR 4100
                         99001 GP38/2 DIESEL   APR 4100
                         E6001 E6     DIESEL   PRR 6200
                         E6900 E6B    DIESEL   PRR 6200
```

Print Locomotive Status

Print version of List Locomotive Status. The data can be filtered by locomotive Type, Railroad, Home Station or any combination of the three. This report gives 'static' details about the locomotives, such as their type, notes, owning railroad and home station.

`PSLOCO (sort[0/1/2/3]);(^loco type^);(^railroad^);(^station^)`

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Locomotive running number order
1 – sorted in Locomotive Type, then Loco number order
2 – sorted by Railroad, then Locomotive Number
3 – sorted by Home Station, then Locomotive Number order

Loco type – filter by type of locomotive (as held on Locomotive Types). optional
Railroad – filter by Owning Railroad. Optional
Home Station – filter by Home Station f locomotive. Optional

```
MOPS/PSLOCO                  ATLANTIC AND PACIFIC              24JUN1972 14:56
CC                        LOCO STATUS SORTED BY LOCO NUMBER         Page:    1

LOCO= TYPE== FUEL= MAINT RPAIR PWR STAT PLACE======================== TRAIN=====

06501 E6      1800    90     0   P 4100                             0
38201 GP38/2  2600    90     0   P 4100                             0
40318 DD40AX  8230   120     0   P 4100                             0
40319 DD40AX  8230   120     0   P 4100                             0
99001 GP38/2  2600    90     0   P 4100                             0
E6001 E6      1800    90     0   P 6200                             0
E6900 E6B     1800    90     0   P 6200                             0
```

Car Classes: defining types of cars

Add a Car Class [Supervisor access only]

Add a Car Class (with a description) which will be used to help group types of cars together.

```
ADCLAS class;class name
```

Class – code size is determined by the setting of **classize**. Required.
Class name – description of the Car Class. Required.

Change a Car Class Description [Supervisor access only]

Allows a Car Class's description to be changed.

```
CHCLAS class;class name
```

Class– code of record to be amended. Required.
Class description – amend the description of the Car Class. Required.

Delete a Car Class [Supervisor access only]

Delete a Car Class from the system. A Car Class cannot be deleted if it refers to a Car Type or is used by a Warehouse as part of its Car ordering requirements.

```
DXCLAS class
```

Class – code of record to be deleted. Required

List Car Classes

List Car Classes available on MOPS.

```
LICLAS (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Code order
1 – sorted in Description order

```
CLA NAME==========================
AUT AUTORACK
BOX BOX CAR
CAB CABOOSE
CBM CENTREBEAM CAR
COL COIL CAR
GON GONDALA
HIC HICUBE
```

Print CarClasses

Print a list of Car Classes to a report for printing.

```
PRCLAS (sort[0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
0 – sorted in Code order
1 – sorted in Description order

```
MOPS/PRCLAS                    ATLANTIC AND PACIFIC                 24JUN1972 14:56
CC                             LIST OF CAR CLASSES                       Page:    1

                        CLA NAME==========================

                        AUT AUTORACK
                        BOX BOX CAR
                        CAB CABOOSE
                        CBM CENTREBEAM CAR
                        COL COIL CAR
                        GON GONDALA
                        HIC HICUBE
```

Car Types: Defining details of Individual Car Types

Add a Car Type *[Supervisor access only]*

Add a Car Type (with a description). A Car Type contains details about individual car types, such as weights, length, etc. The Car Type also defines how an individual car, of that type, will be loaded and unloaded. Note that maintenance details for Car Types are held on the Parameter File: Cars are assumed to have a ‘generic’ maintenance requirement.

```
ADCART car type;type name;length;capacity;unladen weight;oper mode[I/M];
       ^load^;^load^;^car class^
```

Car type– code size is determined by the setting of **cartsize**. Required.
Type name – description of the Car Type Required.
Length – length of the car. Should be consistent in units using other lengths. Required
Capacity – capacity of the car (same units as other weights). Required
Unladen weight – weight of the car. Should be consistent with other weights. Required
Oper mode – Independent or part of Multiple unit (I or M). Required
Load #1 – look-up for loading code required for loading the car. Required
Load #2 – look-up for loading code required for unloading the car. Required
car class – look up for lining the car type to a car class. Required.

Change a Car Type’s details *[Supervisor access only]*

Allows a Car Type’s information to be changed, including loading, unloading and other details.

```
CHCART car type;(type name);(length);(capacity);(unladen weight);
       (oper mode[I/M]);(^load^);(^load^);(^car class^)
```

Car type– code of record to be amended. Required.
Type name – description of the Car Type Optional.
Length – length of the car. Should be consistent in units using other lengths. Optional
Capacity – capacity of the car (same units as other weights). Optional
Unladen weight – weight of the car. Should be consistent with other weights. Optional
Oper mode – Independent or part of Multiple unit (I or M). Required
Load – look-up for loading code required for loading the car. Optional
Load code – look-up for loading code required for unloading the car. Optional
Car class – look up for defining the car type as a car class. Optional.

Delete a Car Type *[Supervisor access only]*

Delete a Car Type from the system. A Car Type cannot be deleted if individual Cars are of this type.

```
DXCART car type
```

Car type – code of record to be deleted. Required

List Car Types

List Car Types available on MOPS.

```
LICART ([0/1])
```

Sort –sort order of data. Optional; defaults to 0 if not entered.
- 0 – sorted in Code order
- 1 – sorted in Description order

```
TYPE NAME========================== LNGTH CAPTY U/WT= O LOAD UNLO CLA
TR50 TRAINCO TYPE 50                   50   139    80 I RAMP RAMP BOX
TR55 TRAINCO TYPE 55                   55   154    95 I RAMP RAMP BOX
G555 JEFFERSON GONDALA                 55   150    70 I HOSE HOSX GON
G220 CHEMICAL GONDALA                  55   125    65 I HOSE HOSX GON
```

Print Car Types

Print a list of Car Types to a report for printing.

`PRCART ([0/1])`

Sort –sort order of data. Optional; defaults to 0 if not entered.

0 – sorted in Code order

1 – sorted in Description order

```
MOPS/PRCART                  ATLANTIC AND PACIFIC              24JUN1972 14:56
CC                            LIST OF CAR TYPES                     Page:    1

     TYPE NAME========================== LNGTH CAPTY U/WT= O LOAD UNLO CLA

     G220 CHEMICAL GONDALA                  55   125    65 I HOSE HOSX GON
     G555 JEFFERSON GONDALA                 55   150    70 I HOSE HOSX GON
     TR50 TRAINCO TYPE 50                   50   139    80 I RAMP RAMP BOX
     TR55 TRAINCO TYPE 55                   55   154    95 I RAMP RAMP BOX
```

Cars: Information about individual coaches and wagons

Add a Car *[Supervisor access only]*

Add a Car. A Car contains information about individual cars. The basic information about the car relates to the owning railroad and its home station; the type of car is also identified, which gives its other details, such as length, weight etc. The current Station of a Car is set to its Home Station, and Maintenance details are obtained from the Parameter file.

`ADCARX car;^car type^;^railroad^;^home station^`

Car – running number of the car, code size is determined by the setting of **carxsize**. Required.

Car type – links to car type details such as length, unladen weight, loading/unloading, etc. Required

Railroad – owning railroad (could be a leasing company: set leasing company as a railroad). Required

Home station – where the car should return to. Required

Change Car Details *[Supervisor access only]*

Change basic information about individual cars. The basic information about the car relates to the owning railroad and its home station; the type of car is also identified which gives its other details, such as length, weight etc.

`CHCARX car;(^car type^);(^railroad^);(^home station^)`

Car – running number of the car that requires to be amended. Required.

Car type – links to car type details such as length, unladen weight, loading/unloading, etc. Optional

Railroad – owning railroad (could be a leasing company: set leasing company as a railroad). Optional

Home station – where the car should return to. Optional

Change Maintenance Details *[Supervisor access only]*

Correcting command. Change maintenance information about an individual car. Allows time to maintenance, or Works time, to be changed outside normal maintenance processes. Although two times are available for input, only the valid time will be accepted (eg if in works, then the works time will be processed).

`MAINTC car;(time to maint^);(^works time^)`

Car – running number of the car, code size is determined by the setting of **carxsize**. Required.

Time to maint – number of days until car requires maintenance. Optional

Works Time – time in hours that the car will be in works. optional

Allocate Car to Block
Allocates a car to a block. If the block does not exist, then a new block with that block reference will be created (the block reference should be 5 characters or fewer). From that point on, any operation performed on any car in that block will result in that same operation being carried out on all cars in that block (the intended use is as a shortcut to moving groups of cars as a unit, rather than having to deal with each car independently).

```
ACARXB car;block
```
Car – running number of the car to be attached to the block. Required
Block – new or existing block reference. Required

Remove Car from Block
Removes a specific car from a block. If this is the last car for which that block reference exists, then the block reference will no longer exist, although it could, if required, be re-used.

```
XCARXB car
```
Car – running number of the car to be detached from the block. Required

Allocate Car to Set
Allocates a car to a set. The set must have previously been created on a powered unit (loco). The car will no longer be available for moving, reporting etc and will move with the powered part of the set. Used for multiple unit working.

```
ACARXS car;^loco^
```
Car – running number of the car to be allocated to the set. Required.
Loco – loco number to which the unpowered part of the set will be attached. Required.

Remove Car from Set
De-allocates a car from an existing set. The car can only be de-allocated from the set when the locomotive is not on a train. The car takes the same location as the current location of the locomotive.

```
XCARXS car
```
Car – running number of the car to be detached from the set. Required.

Locate Car at Station *[Supervisor access only]*
Correcting command. This command is used when a car has been mis-reported and the current station needs to be updated for the car. The normal mechanism for a car to change stations will be on a Train.

```
CARXAT car;^station^
```
Car – running number of car to be re-located at a different station. Required.
Station – station reference at which the car should be located.

Spot Car at Place
This moves a Car that is at a Station to a specific Place or Industry within that Station. This then allows any specific actions at that Place (Loading, Unloading, Maintenance, Cleaning) to start taking place on the Car.

```
CARXSP car;^place^
```
Car – running number of car to be spotted. Required.
Place – place at which the car is to be spotted. Required.

Set Car to Empty/Clean *[Supervisor access only]*

Correcting command. This command sets the status of a car to empty and clean. Used to correct the status of the car. The normal mechanism for Cars to Empty will be at an Industry, and to be cleaned at a Cleaning Place.

```
CLEANX car
```

Car – running number of car to be set to clean and empty. Required.

Delete Car *[Supervisor access only]*

Removes a Car from MOPS. Not allowed if the car is being processed at a place or is on a train, or is part of a block or a set.

```
DXCARX car
```

car – running number of car to be removed from MOPS. Required

Report Cars by Status

List cars and their current location, including whether empty, clean, or in maintenance. Also indicates whether car is attached to a block. Can be filtered by car class and/or station. Cars which are attached to a Set are not listed.

```
LACARS (^car class^);(^station^)
```

Car class – class of car for which filtering is required. Optional
Station – station for which filtering is required. Optional

```
RAI CAR==== CLA STATION/TRAIN PLACE============== COM DESTINATIO C BLOCK MAINT
PRR 0005042 BOX 1000 SPRNGFLD 0                                             90
PRR 0005043 BOX 1000 SPRNGFLD 0                                             90
PRR 0005044 BOX 1000 SPRNGFLD 0                                             90
PRR 0005045 BOX 1000 SPRNGFLD 0                                             90
PRR 0005046 BOX 1000 SPRNGFLD 0                                             90
PRR 0005047 BOX 1000 SPRNGFLD 0                                             90
```

List Cars

List cars and their status including allocation to any block or set.

```
LICARS (sort[0/1/2/3]);(^car type^);(^railroad^);(^station^)
```

Sort – sort option, defaults to 0 if not supplied;
- 0 – sorted in Car running number order
- 1 – sorted in Railroad / Car running number order
- 2 – sorted in Car Class / Car order
- 3 – sorted in Station / Car order

Type – Type of car for which filtering is required. Optional
Railroad – railroad ownership of car. Optional
Station – station at which Cars are currently located. Optional

```
CAR==== TYPE CLA RAI HOME============================== MAINT RPAIR BLOCK===== SET=
0005042 TR50 BOX PRR 1000(SPRINGFIELD)                     90     0             0
0005043 TR50 BOX PRR 1000(SPRINGFIELD)                     90     0             0
0005044 TR50 BOX PRR 1000(SPRINGFIELD)                     90     0             0
0005045 TR50 BOX PRR 1000(SPRINGFIELD)                     90     0             0
0005046 TR50 BOX PRR 1000(SPRINGFIELD)                     90     0             0
0005047 TR50 BOX PRR 1000(SPRINGFIELD)                     90     0             0
```

Report Unallocated Empty Cars

Lists Cars that have not been allocated to fulfil customer requirements and are therefore available to be assigned to any outstanding Empty Orders. Cars not at a Station are not reported.

`LMTCAR (^area^);(^station^);(^car class^)`

Area – provides a list of cars that are available at Stations in that Area. Optional
Station – provides a list of cars available at that Station. Optional.
Car Class – filters the list to include Cars of that Car Class only. Optional.

```
RAI ARE STAT NAME==== PL TYPE CAR==== C BLOCK    LOAD UNLO
PRR SPR 1000 SPRNGFLD    TR50 0005042            RAMP RAMP
PRR SPR 1000 SPRNGFLD    TR50 0005043            RAMP RAMP
PRR SPR 1000 SPRNGFLD    TR50 0005044            RAMP RAMP
PRR SPR 1000 SPRNGFLD    TR50 0005045            RAMP RAMP
PRR SPR 1000 SPRNGFLD    TR50 0005046            RAMP RAMP
PRR SPR 1000 SPRNGFLD    TR50 0005047            RAMP RAMP
```

Allocate Empty to Order

Allocates a Car to a particular Empty Order. If a Car is associated with a Block, then all the cars of that Block are associated with that Order. If the number of cars assigned to the Order exceeds that required by the order, then the assignment will be rejected. Up to 5 cars may be input to the order at any one time – the number of cars actually assigned will depend on the number of cars associated with all the referenced blocks. Cars cannot be assigned if loaded (or if the order requires clean cars and the car is not clean). The car cannot be assigned if teh Loading and/or unloading codes are not compatible with the originitaing industry and/or Destination Industry.

`MTYORD order;car;(car);...` ***`[up to 5 cars]`***

Order – Empty Car Request reference id. Required.
Car – running number of car being assigned. Up to 5 car numbers may be input. Required.

Report Empties En Route

Lists Cars that have been assigned to fulfil an Empty Order.

`LONWAY (^station^);(^industry^)`

Station – limits output to Cars that are currently at that Station. Optional.
Industry – limits list to Cars that are on their way to that Industry. Optional.

```
RAI CAR===== STATION/PLACE===== DESTINATN= COMMODITY=================== CURRENT
APR 7000     7000/02 ALBANYHOSE HRTFRDHOSE 315  INK PRODUCTS              4100
APR 7000     7000/02 ALBANYHOSE HRTFRDHOSE 315  INK PRODUCTS              4100
```

Print Cars

Prints a list of all cars in Railroad/Car Running Number order. List may be filtered by Car Class and/or Station.

```
PRCARS (sort[0/1/2/3]);(^car class^);(^railroad^);(^station^)
```

Sort – sort option, defaults to 0 if not supplied;
- 0 – sorted in Car running number order
- 1 – sorted in Railroad / Car running number order
- 2 – sorted in Car Class / Car order
- 3 – sorted in Station / Car order

Car Class – show only these Car Classes when printing. Optional
Railroad – show only these cars that belong to this railroad. Optional.
Station – show only these cars at that Station when printing. Optional

```
MOPS/PRCARS                   ATLANTIC AND PACIFIC                24JUN1972 14:56
CC                                LIST OF CARS                         Page:    1

RAI CAR==== CLA STATION/TRAIN PLACE================ COM DESTINATIO C BLOCK MAINT

PRR 0005042 BOX 1000 SPRNGFLD 0                                               90
PRR 0005043 BOX 1000 SPRNGFLD 0                                               90
PRR 0005044 BOX 1000 SPRNGFLD 0                                               90
PRR 0005045 BOX 1000 SPRNGFLD 0                                               90
PRR 0005046 BOX 1000 SPRNGFLD 0                                               90
PRR 0005047 BOX 1000 SPRNGFLD 0                                               90
```

Routes and Schedules

Routes – Identifying where the Trains will run

Add Route *[Supervisor access only]*

Schedules are based on Routes: A Route record defines a code, name and Default Direction for a Route. Sections (which are effectively ‘links’ between Stations) are used to build up a Route – see information on Sections following this. The Route is initially set to Incomplete.

```
ADROUT route;route name;default direction[E/W/N/S/U/D]
```

Route – route code. Size of field is defined by **routsize**. Required
Route name – name of route. Required.
Default Direction – main or default direction of route. Required.
- E – East. Schedules can be established on this Route running East or West.
- W – West. As above.
- N – North. Schedules will be North/South
- S – South. As above
- U – Up. UK Term. Schedules can be established running Up/Down on this Route.
- D – Down. See above

Change Route *[Supervisor access only]*

Allows a Route’s name to be changed. The default direction for the Route cannot be changed.

```
CHROUT route;route name
```

Route – code of route to be amended. Required.
Route name – name of route to be amended. Required.

Delete Route *[Supervisor access only]*

Delete's a Route from the system, along with any associated Sections and/or Instructions. A Route cannot be deleted if it is in Published status.

`DXROUT route`

Route – code of route to be deleted. Required.

Validate Route *[Supervisor access only]*

This validates that the various sections of the route are contiguous. A Route requires to be Validated before it can be published and made available for setting Schedules against.

`VALIDR route`

Route – code of Route to be validated. Required.

Publish Route *[Supervisor access only]*

This changes the status of the Route so that Schedules can be set against the Route.

`PUBLSH route`

Route – code of route to be published. Required.

Set Published Route to Draft *[Supervisor access only]*

This takes a previously Published Route and sets it to draft status to allow further amendment to the Route. A Route cannot be set to draft Status if it has any Schedules running against it.

`UNPUBL route`

Route – Code of route to be set to Draft status.

List All Routes

Lists all routes on MOPS in Route code order. Includes default direction, status and arriving and departing stations.

`LIROUT`

```
RO NAME=========================  DIR STATUS DEPART======= ARRIVE=======
01 PITTSFIELD-SCHODACK             E  INCMPL 6200 PITTSFLD 7000 SCHODACK
R1 ROUTE 1                         W  PUBLSH A STAX_A      E STAX_E
R2 ROUTE 2                         W  INCMPL A STAX_A      F STAX_F
R3 ROUTE 3                         W  INCMPL B STAX_B      E STAX_E
02 RUSSELL ROAD-SCHODACK           E  INCMPL 2100 RSSLROAD 7000 SCHODACK
```

Print All Routes

Prints all Routes in Route Code order. Includes Status and Arrival/Departure stations.

`PRROUT`

```
MOPS/PRROUT                    ATLANTIC AND PACIFIC              25JUN1972 14:22
CC                            SUMMARY LIST OF ROUTES                  Page:    1

    RO NAME=========================  DIR STATUS DEPART======= ARRIVE=======

    01 PITTSFIELD-SCHODACK             E  INCMPL 6200 PITTSFLD 7000 SCHODACK
    02 RUSSELL ROAD-SCHODACK           E  INCMPL 2100 RSSLROAD 7000 SCHODACK
    03 SPRINGFIELD-SCHODACK            E  PUBLSH 1000 SPRNGFLD 7000 SCHODACK
```

Sections – Identifying the Parts of the Route

Add Route Section *[Supervisor access only]*

Add a Route section to a Route. Only allowed when the Route is in draft status. To complete a Route, all sections should be contiguous: apart from the departing and arrival station, an arrival station for one section must be the same as the departure station for the next section when the sections are arranged in section order. The section numbers need not be consecutive but do need to be in order.

```
ADSECT ^route^;section;^from^;^to^
```

Route – route code for which the section is being added. Required.
Section – sequence or order number for the section
From - the starting point station for the section. Required.
To– the ending point station for the section. Required.

Delete Route Section *[Supervisor access only]*

Deletes a section from a Route, if the route is in Draft or Incomplete Status. If the Route is in Draft status it is set to Incomplete and will require to be re-validated.

```
DXSECT section id
```

Section id – the unique system id for the section to be deleted. Required.

List Sections for Route

Lists the status of a given route and the associated sections in order.

```
LSROUT route
```

Route – code of route to be listed. Required

```
ROUTE:01 PITTSFIELD-SCHODACK DIRN:EAST STATUS:INCOMPLETE
ID=== SECTION= DEPARTING==== ARRIVING=====
   10       10 6200 PITTSFLD 6100 VALLEYYD
   11       20 6100 VALLEYYD 6000 FORSTJCN
   12       30 6000 FORSTJCN 5050 PFDAPRIC
   13       40 5050 PFDAPRIC 5000 FELHAMJN
   14       50 5000 FELHAMJN 7000 SCHODACK
```

List Detail for Selected Route

Lists full detail for a route, including any instructions for stations or for the route itself.

```
LDROUT route
```

Route – code of route to be listed. Required

```
ROUTE: 01 PITTSFIELD-SCHODACK  (ROUTE STATUS:INCOMPLETE)
       DIRECTION: EAST
 -  RESTRICTED SPEED AND WEIGHT OVER MOHAWK BRIDGE

DEPARTING==============================  STATION INSTRUCTIONS ===============
6200 PITTSFIELD
6100 VALLEY YARD                         STOP AT WESTFIELD FOR INSTRUCTIONS
6000 FORREST JUNCTION
5050 PITTSFIELD/APR I/C
5000 FELHAM JUNCTION                     MAX SPEED 30MPH THROUGH JUNCTIONS
                                         NO ACCESS TO YARD FROM FAST LINES
7000 SCHODACK                            MAX SPEED 25MPH IN STATION
                                         PASSENGER TERMINAL
```

Print Detail for Selected Route

Prints details for a given route, including any Station or Route instructions.

`PDROUT route`

Route – code of route to be printed. Required.

```
MOPS/PDROUT                    ATLANTIC AND PACIFIC                25JUN1972 14:22
CC                                ROUTE LISTING                         Page:    1

   DEPARTING======================== INSTRUCTIONS =========================

   ROUTE: 01 PITTSFIELD-SCHODACK (ROUTE STATUS:INCOMPLETE)
          DIRECTION:EAST
    - RESTRICTED SPEED AND WEIGHT OVER MOHAWK BRIDGE

   10 PITTSFIELD
   20 VALLEY YARD                    STOP AT WESTFIELD FOR INSTRUCTIONS
   30 FORREST JUNCTION
   40 PITTSFIELD/APR I/C
   50 FELHAM JUNCTION                MAX SPEED 30MPH THROUGH JUNCTIONS
                                     NO ACCESS TO YARD FROM FAST LINES
   7000 SCHODACK                     MAX SPEED 25MPH IN STATION
                                     PASSENGER TERMINAL
```

Schedules – Identifying what Trains will be required

Add Schedule *[Supervisor access only]*

A Schedule determines when trains will run. Initially, a schedule is given a code, name and a class (or priority). The days on which the schedule will run are also given (these are added as days of the week, with an eighth character indicating whether the train runs on MOPS Calendar holidays). The direction is also required, which must either agree with or be opposite to the default direction of the Route. The times of the Schedule are given later.

`ADSCHD schedule;^route^;name;class;rundays[12345678];direction[N/S/E/W/U/D]`

Schedule – code of new schedule. Required.
Route – route code for which the schedule applies. Required.
Name –any particular name or identifier for the Schedule. Required
Rundays – days on which schedule will run, Mon=1, Hols=8. *If not running enter a . (dot) at that position. Note that this is position ie Wednesday must be at position 3, etc*. Required.
Direction – must equal or oppose the default direction of the Default Direction. Required

N – if default direction of Route is N or S
S – see above
E – if default direction of Route is E or W
W – see above
U – if default direction of Route is Up or Down
D – see above

Change Schedule *[Supervisor access only]*

Change either the name, class or run days of the Schedule (Once a Schedule has been set up, it cannot be allocated to a new Route nor have a change of default direction).

`CHSCHD schedule;(name);(class);(rundays[12345678])`

Schedule – code of schedule to be changed. Required.
Name –any particular name or identifier for the Schedule. Optional
Rundays – days on which schedule will run, Mon=1, Hols=8. *If not running enter a . (dot) at that position. Note that this is positional ie Wednesday must be at position 3, etc*. Optional..

Delete Schedule *[Supervisor access only]*

Deletes an Inactive Schedule from MOPS, including all associated Timings and Instructions.

```
DXSCHD schedule
```

Schedule – code of Schedule to be deleted. Required.

Set Schedule Inactive *[Supervisor access only]*

Sets an Active Schedule to Inactive (normally to allow it to be deleted). Not allowed if trains are running against the Schedule.

```
XCTIVE schedule
```

Schedule – code of schedule to be set to Inactive. Required.

Activate Schedule *[Supervisor access only]*

Sets a Schedule to Active. Validate that a schedule has a complete set of times for each Station and set to Active status. Only Active Schedules can have trains running on them.

```
ACTIVE schedule
```

Schedule – code of schedule to be validated and activated. Required.

Cancel Active Schedule

Cancels an active schedule for that day. This means that any train that would be due to run against that schedule will no longer be able to do so. The schedule will be available again on the next day it is due to run.

```
CXSCHD schedule
```

Schedule – id of schedule to be cancelled

List all Schedules

Provides a summary of all Schedules on MOPS, including details of Status, days running, Direction, Class, Time Departing (if available) and the Origin and Destination of the Schedule. Listing can be restricted by Status, or be limited to a single day.

```
LISCHD (status);(runday[1/2/3/4/5/6/7/8])
```

Status – restrict list of Schedules to that Status. Optional.
Runday – restrict list to given day of week. Optional.

```
SC RO NAME========================== STATUS== RUN DAYS D C =DEP ORIG DEST
#4 02 EASTERN LOCAL (WEST)           INACTIVE 1...5..8 W 3      7000 2100
22 03 SPRINGFIELD-SCHODACK           ACTIVE   12345678 E 2 0510 1000 7000
#1 R1 THE ALPHABET FLIER             INACTIVE 1.3.56.8 W 1 1000 A    E
#6 02 EASTERN LOCAL #2               INACTIVE 12345..8 E 2 1053 2100 7000
#2 R1 THE EASTBOUND FLIER            INACTIVE 1.3.56.8 E 1 1200 E    A
#3 R1 THE WESTBOUNDER                ACTIVE   1.3.56.8 W 1 1800 A    E
23 03 EVENING RUN                    RUNNING  12345678 E 2 1830 1000 7000
```

Print All Schedules

Prints a summary of all Schedules, including details of Status, days running, Direction, Class, Time Departing (if available) and the Origin and Destination of the Schedule. Listing can be restricted by Status, or be limited to a single day.

```
PRSCHD (status);(runday[12345678])
```

Status – restrict list of Schedules to that Status. Optional.
Runday – restrict list to given day of week. Optional.

```
MOPS/PRSCHD                    ATLANTIC AND PACIFIC                   25JUN1972 14:22
CC                              LIST OF SCHEDULES                          Page:    1

        SC RO NAME=========================  STATUS== RUN DAYS D C =DEP

        #4 02 EASTERN LOCAL (WEST)           INACTIVE 1...5..8 W 3
        22 03 SPRINGFIELD-SCHODACK           ACTIVE   12345678 E 2 0510
        #1 R1 THE ALPHABET FLIER             INACTIVE 1.3.56.8 W 1 1000
        #2 R1 THE EASTBOUND FLIER            INACTIVE 1.3.56.8 E 1 1200
        #3 R1 THE WESTBOUNDER                ACTIVE   1.3.56.8 W 1 1800
        23 03 EVENING RUN                    RUNNING  12345678 E 2 1830
```

List Active and Running Schedules

Lists all Schedules which are either due to run or have a scheduled train running against them for the current day (ie completed or cancelled trains are not shown). If a train has departed, then the Status is shown as TRAIN*; if a train has been assembled (albeit possibly onlu a locomotive having been assigned) but has not yet departed then it is shown as –READY.

```
LSSCHD
SCHE C RO ORIG DEST TIME STATUS TRAIN
#401 1 04 1200 7300 0800 TRAIN* #401
#402 1 04 7300 1200 0800 TRAIN* #402
#82  3 08 4250 2060 0826 TRAIN* #82
#404 1 04 7300 1200 1300 -READY #404
#403 1 04 1200 7300 1400 -READY #403
#86  3 08 4250 2060 1543 ACTIVE
```

Timings – Adding the detail to the Timetable

Add Schedule Timings *[Supervisor access only]*

Adds Timings to an indicated schedule. This command operates in a different way to other commands. Each Departure and Arrival for each section will be prompted (to quit early, enter 'x' at a prompt). Times will be validated for sequence (schedules going over midnight are accommodated). Through trains at particular points (eg Junctions) should have the same Arrival and Departure times. A complete example of prompts and inputs are shown below. If the sequence is ended early, then completed section timings will be saved.

```
ADTIMS schedule
```

Schedule – code for which timings are to be added

```
SCHEDULE ENTRY MODE: ENTER TIME HHMM OR <X> TO QUIT
TIME DEPARTING 2100 RSSLROAD >1053
TIME ARRIVING  2000 RSLRDJCN >1105
TIME DEPARTING 2000 RSLRDJCN >1106
TIME ARRIVING  3000 WSTFIELD >1314
TIME DEPARTING 3000 WSTFIELD >1622
TIME ARRIVING  4000 STCKBRDG >1700
TIME DEPARTING 4000 STCKBRDG >1700
TIME ARRIVING  4100 STKBRGYD >1812
TIME DEPARTING 4100 STKBRGYD >2011
TIME ARRIVING  4200 STKBRGJN >2048
TIME DEPARTING 4200 STKBRGJN >2048
TIME ARRIVING  5000 FELHAMJN >2218
TIME DEPARTING 5000 FELHAMJN >2218
TIME ARRIVING  7000 SCHODACK >2347
UPDATE OF SCHEDULE TIMINGS FOR #6 COMPLETED
```

Copy Schedule *[Supervisor access only]*

Copy an existing Schedule in order to create a new Schedule. The new schedule will contain the same information as the old schedule but will carry a new reference and a new name. In addition, all timings on the new route will be offset by the time difference between the start time of the old route and the start time of the new route.

```
CPSCHD ^old schedule^;new schedule;name;start time
```

Old – code of old schedule from which the new schedule will be copied. Required
New – code for new schedule to be created. Required.
Name – name of new schedule. Required.
Start – start time of new schedule (other timings will be offset by old-new difference). Required.

Change Schedule Timings *[Supervisor access only]*

Change a section's departure and arrival times. Can't be amended if a train is already running on the Schedule.

```
CHTIMS schedule;section;depart;arrive
```

Schedule – code of schedule whch requires to be amended. Required
Section – section code (of Route) which requires to be amended. Required
Depart – new departure time for the station (time entering section). Required.
Arrive – arrival time for the station (time leaving section). Required.

List Timings for Selected Schedule

Lists timings including Instructions for the given Schedule, and including any additional instructions for Stations and the Route.

`TIMING schedule`

Schedule – code of Schedule for which timing is required

```
SCHEDULE: #3 THE WESTBOUNDER  (SCHEDULE STATUS:INACTIVE)
       DIRECTION: WESTBOUND
 -  CONVEY SLEEPING CARS FROM STATION C

STAT NAME==== TYP =ARR =DEP INSTRUCTIONS =========================
A    STAX_A   STN      1800
C    STAX_C   STN 2050 2225
D    STAX_D   STN 0000 0000 NO ACCESS TO ROAD 1 FOR EASTBOUND TRAINS
E    STAX_E   STN 0232
 ** END OF DATA:3  RECORDS DISPLAYED **
```

Print Timings for Selected Schedule

Print timings including status and Instructions (for the Schedule, Route or Stations).

`PRTIMS schedule`

Schedule – code of Schedule for which timing is required

```
MOPS/PRTIMS                  ATLANTIC AND PACIFIC                25JUN1972 14:22
CC                             TIMETABLE FOR #3                       Page:    1

      STAT NAME==== TYP =ARR =DEP INSTRUCTIONS =========================

      SCHEDULE: #3 THE WESTBOUNDER  (SCHEDULE STATUS:ACTIVE)
             DIRECTION: WESTBOUND
-  CONVEY SLEEPING CARS FROM STATION C

STAT NAME==== TYP =ARR =DEP INSTRUCTIONS =========================
A    STAX_A   STN      1800
C    STAX_C   STN 2050 2225
D    STAX_D   STN 0000 0000 NO ACCESS TO ROAD 1 FOR EASTBOUND TRAINS
E    STAX_E   STN 0232
 ** END OF DATA:3  RECORDS DISPLAYED **
```

List Timings for Selected Schedule

Lists timings for the selected schedule. This is shown in a different way to normal schedule displays, as it shows the timings by section rather than by station – that is, time starting a section, and time finishing a section. This reflects how schedules are loaded into MOPS and is useful for reviewing schedules priot to editing.

`LDTIMS ^schedule^`

Schedule – schedule for which the section timings are to be shown

```
SECTION=== DEPA =DEP ARRI =ARR
12         2100 0730 2050 0755
14         2050 0755 2000 0819
15         2000 0820 3000 0840
16         3000 0842 4000 0852
17         4000 0852 4100 0900
18         4100 0900 4200 0940
20         4200 0940 5000 1006
26         5000 1006 7000 1125
```

Instructions – Information for Stations, Routes and Schedules

Add Instruction *[Supervisor access only]*

Adds special instructions for either Stations, Routes or Schedules (as required). This data will be displayed on some enquiries and reports for information purposes.

```
ADINST element[ROUT/SCHD/STAX];route/schedule/station;instruction
```

Element – indicate the element being amended. Required.

- Rout – instruction is for a Route
- Schd – instruction is for a Schedule
- Stax – instruction is for a Station

Route/Schedule/Station- code for required element for which instruction being added. Required.
Instruction – detailed instruction to be added. Required.

Delete Instruction *[Supervisor access only]*

Deletes the Instruction. Note that Instructions are automatically deleted if the corresponding Station, Route or Schedule is also deleted.

```
DXINST id
```

Id – Instruction id to be deleted.

Trains: Creating, Building, Running and Terminating Trains

Set Unscheduled Train

Creates an Unscheduled train. This train will have no timings nor any indicated destination. The train will start from the station at which the Locomotive is situated. The locomotive must be in power and be capable of operating independently or as part of a set. The progress of an unscheduled train is dictated by the reporting of the train.

```
UTRAIN train;^locomotive^
```

Train – train reference. Required.
Locomotive – power unit to be added to a train. Required.

Set Scheduled Train

Creates a Scheduled Train. A Train is formed when a Locomotive is attached to the Schedule. This train will be monitored against previously published time. The train will follow the route laid down in the schedule. The Schedule will be updated to indicate that a train is running against that schedule. The locomotive must be in power and be capable of operating independently or as part of a set.

```
STRAIN schedule;(train);^locomotive^
```

Schedule – code of schedule for which train is being run. Required.
Train – train code. If not entered, the schedule code will be used as the train code. Optional.
Locomotive – power unit to be added to the train. Required.

Set Extra Train

Creates an Extra Train. A Train is formed when a Locomotive is attached to the Schedule. The train will run against the schedule to which it has been applied. A specific train code (not equal to the schedule code) must be entered. It will be tracked against the original schedule. The locomotive must be in power and be capable of operating independently or as part of a set.

```
STRAIN schedule;train;^locomotive^
```

Schedule – code of schedule against which the train will run. Required
Train – new train code (not equal to any existing schedule or train code). Required.
Locomotive – power unit being added to the Train. Required.

Start Train at Later Origin

Allows a Scheduled or Extra train to be started at a station other than its original departure station. The new starting location must be part of the original schedule. The starting location will be the location of the locomotive.

```
LTRAIN type[S/E];schedule;(train);^locomotive^
```

Type – nature of the train being created. Required
S – Scheduled train
E – extra train
Schedule – code of schedule for which the train is being created. Required
Train – code of train being created. For Extract trains, must not equal the schedule code. *For Scheduled trains, can be left blank (will default to Schedule code)*. Optional.
Locomotive – locomotive being used to create the train. Required.

Terminate Train

Terminates train at a destination. All locomotives and cars will be located at that Station and the train will no longer exist. If the train is Scheduled or is an Extra and is being stopped short of its original destination, then confirmation must be required by entering YES as the second data parameter.

```
TTRAIN train;(confirm[/YES])
```

Train – train code of train being terminated. Required.
Confirm – if short of destination, then confirm with YES.

Allocate Loco to train

Adds another power unit to a train. The locomotive must be at the same location as the train (the train must previously have been reported at that station).

```
ALOCOT locomotive;^train^
```

Locomotive – running number of locomotive to be added to train. Required
Train – train code for the train to which the locomotive is being added.

Remove Loco from Train

Removes a power unit from a train. If this is the last locomotive on the train, it will still continue to exist until terminated (although a loco is required to create a train). The locomotive will then be placed at the current Station of the Train, which must have been reported at that Station.

```
XLOCOT locomotive
```

Locomotive – running number of locomotive to be removed from train

Allocate Car to Train

Adds a car (or block of cars) to a train. The car(s) must be at the same location as the train (the train must previously have been reported at that station).

```
ACARXT car;^train^
```

Car – running number of car to be added. If the car is part of a block, the block is added. Required
Train – train code for the train to which the car(s) is/are being added.

Remove Car from train

Removes a Car from a Train. The train must previously been reported at that Station. If the Car is part of a block, then the entire block will be removed from the Train. The Car(s) will assume the Station as their current location.

```
XCARXT car
```

Car – car id of car being removed from train. Will apply to block. Required.

Report Train

Indicate progress of a train. A Train does not necessarily have to be reported at all points, intermediate points that are not indicated are ignored. For Unscheduled Trains, this command adds the progress of the route; for other trains the actual/expected times at that location will be displayed for information.

```
REPORT train;update[DEP/ARR/THR];^station^;(time)
```

Train – train code of train being reported. Required
Update – nature of update. Required
- DEP – time train departed station
- ARR – time train arrived at the station
- THR – time train arrived/departed at the station

Station – station for which report is being made. Required
Time – time to be reported (defaults to current MOPS time). Optional

Line-up Enquiry

Reports trains en route to given Station, including a breakdown of the consist, scheduled arrival time, where it has jsut left and the original planned route.

```
LINEUP ^station^; (direction)
```

Station – code of station to show arriving trains. Required.
Direction – direction of trains. Optional

```
LINE UP ENQUIRY FOR 7000 SCHODACK
TRAIN= DUE= EST= FROM ORIG DEST C #LOCO #CARS #LOAD #MTY= WGHT= LNGTH
23     0435       5000 1000 7000 2     1     1     0     1    80   109
```

List Running Trains

Shows all running trains, including status of train, Route, schedule, direction, origin, destination, last reported and expected/actual time at last report. The Next station and timetables arrival time is also given; a suffix of '*' indicates a train terminating at that location; a suffix of '+' indicates a timetabled stop (ie departure time is different to arrival time)

`TRAINS (^route^);(direction[N/S/E/W/U/D])`

Route – limit reported train to given route. Optional

Direction – limit reported trains to a given direction – Optional

```
TRAIN= C TYPE RO SCHE D ORIG DEST     LAST DUE. ACT. NEXT EXP.
#402   1 SCHD 04 #402 E 7300 1200 DEP 5000 1119 1119 4200 1222
#401   1 SCHD 04 #401 W 1200 7300 DEP 3000 1145 1145 4000 1223 +
#82    3 SCHD 08 #82  E 4250 2060 DEP 2000 1158 1158 2050 1243
#404   1 SCHD 04 #404 E 7300 1200                    DEP: 1300
#403   1 SCHD 04 #403 W 1200 7300                    DEP: 1400
```

List Consist for Train

Lists summary or detailed information for a train, including power units and car information.

`LICONS train;(report[F])`

Train – train code for report. Required

Report – indicate whether full listing required. Optional.

F – display full listing

<u>Summary:</u>

```
TRAIN:23 CLASS:2 TYPE:SCHD ROUTE:03 SCHEDULE:23 DIRECTION:E
ORIG:1000 SPRNGFLD DEST:7000 SCHODACK ARR AT:7000 SCHODACK
DUE:0435 ACT:2200 LOCOS:0 LOAD:0 MTY:0 L:0 W:0
```

<u>Full</u>

```
TRAIN:#95 CLASS:1 TYPE:SCHD ROUTE:04 SCHEDULE:#95 DIRECTION:E
ORIG:1000 SPRNGLFD DEST:7000 SCHODACK     AT:1050 SPRFLDYD
DUE:      ACT:      LOCOS:1 LOAD:0 MTY:2 L:169 W:130
RAI LOCO POWER= TYPE== HAULS MODE== FUEL= HOME        LNGTH WGHT=
APR 1011        DIESEL  5000          2600 1050          59     0
--- CAR===== DEST ======== CUSTOMER   BLOCK  CLA COMM ----- -----
APR 0050317                                  TNK          55    65
APR 0050318                                  TNK          55    65
```

Report Consist for Train

Prints summary or detailed information for a train, including power units and car information.

`PRCONS train;([F])`

Train – train code for report. Required

Report – indicate whether full listing required. Optional.

F – display full listing

```
MOPS/PRCONS                   ATLANTIC AND PACIFIC             25JUN1972 14:22
CC                            CONSIST FOR TRAIN 23                  Page:    1

        TRAIN:23 CLASS:2 TYPE:SCHD ROUTE:03 SCHEDULE:23 DIRECTION:E
        ORIG:1000 SPRNGFLD DEST:7000 SCHODACK ARR AT:7000 SCHODACK
        DUE:0435 ACT:2200 LOCOS:0 LOAD:0 MTY:0 L:0 W:0

        RAI LOCO= POWER= TYPE== HAULS MODE== FUEL= HOME     LNGTH WGHT=

        RAI CAR==== DEST ======== CUSTOMER   BLOCK  CLA COM LNGTH WGHT=
```

Users: maintaining access to the system

Add a User *[Supervisor access only]*

Adds a new user to MOPS, with a user id, name and type of access allowed. Type of Access determines what the user can do within MOPS.

```
ADUSER user;user name;(user type[S/N/R])
```

id – access id of user. Up to 8 characters. Required.
Name – describes the User. Required.
Access – what the user can do in the system. Required
S-Supervisor access. Can use all commands.
N-Normal access; operational running updates
R-read only; listing and reporting only

Change a User's name or access level *[Supervisor access only]*

Amend a user's name or access to the system

```
CHUSER user;(user name);(user type[S/N/R])
```

id – access id of user being amended. Required.
Name – describes the User. Optional
Access – what the user can do in the system. Optional
S-Supervisor access. Can use all commands.
N-Normal access; operational running updates
R-read only; listing and reporting only

Delete a User *[Supervisor access only]*

Delete a User from the system

```
DXUSER user
```

id – id of user to be deleted. Required

Disable/Enable User's access to MOPS *[Supervisor access only]*

Enable or disable a User's access to the system (enables if disabled; disabled if already enabled).

```
EDUSER user
```

id – id of user to be switched between enabled/disabled or vice versa

Reset a User's Password *[Supervisor access only]*

Resets a User's Password to the system default (PASSWORD). The user will be prompted at sign-on to change the password.

```
RESETP user
```

id – id of user for whom the password is being reset

Allow a User to change their own Password

Allows a user to change their own password

```
CHPASS
```

List Users (to screen, 20 records at a time)

Lists users to the screen. There are no sort or filter options.

```
LIUSER
```

```
USERID== NAME======================= TYP ACT DIS
BF       BRIAN                         N   N   N
CC       COLIN CREWE                   S   Y   N
```

List Users (to a report, for sending to a printer)

Lists users to a printed report. There are no sort or filter options

.PRUSER

```
MOPS/PRUSER                    ATLANTIC AND PACIFIC               24JUN1972 13:54
CC                         LIST OF USERS IN NAME ORDER                 Page:    1

               USERID== NAME=========================== TYP ACT DIS

               BF       BRIAN                            N   N   N
               CC       COLIN CREWE                      S   Y   N
```

Background Tasks

Z000: Advance MOPS Clock. This simply advances the MOPS clock by one minute. The speed at which it does this depends on the setting CSPEED. MOPS checks every five seconds as to whether to advance its own clock.

Z001: Generate Production. Each hour, MOPS creates the appropriate quantity of goods at the warehouse. If the amount of goods breaches the threshold, MOPS will order cars of the given type. If subsequent thresholds are breached, then additional cars will be ordered. If the warehouse reaches maximum capacity, then production ceases until the backlog is cleared.

Z002: Maintain Loco Countdown. Each day, each locomotive is 'counted down' until it reaches zero: before it reaches zero, the locomotive should be put into maintenance to restore the counter to the maximum value for that locomotive type. MOPS doesn't stop running the locomotive if it falls to or below zero, it just produces a lot of messages.

Z003: Generate Flash Message. Flash Messages can be automatic or user-generated: this mechanism makes sure that everyone is set up to get a message.

Z004: Report Flash Messages. This mechanism makes sure that the messages are delivered.

Z005: Waybill. This reads all cars at a location. If all the cars are loaded, then a Waybill is raised and a message issued indicating that the Cars are ready to depart. If all cars have been delivered to the destination and the cars have been unloaded, a message is issued indicating that the cars have now been finished with, and the cars can go back to their home location.

Z006: Expire Flash Messages. Any messages over 24 MOPS Hours old are deleted.

Z008: Set Schedules to Running. This runs once a day and identifies those schedules that are due to run that day.

Z011: Maintain Car Countdown. As Z002, but for Cars.

Z012: Works Cars. When a Car is spotted at a Car Maintenance place, an hourly countdown begins while the car is being maintained. When the counter reaches zero, a message is sent out and the Car Countdown for the car is restored to the value set on the parameter file.

Z013: Select Car for Loading. If an empty car has been spotted at an industry, then the car will be selected for loading. Only one car can be loaded at a time, and this routine will automatically select the next empty car if one is available.

Z014: Set Locomotive Fuel Usage. If a loco is on a train, then the fuel on-board is depleted. If it reaches zero, warning messages will be issued. Re-locating a loco automatically reduces on-board fuel by 25%.

Z015: Car Loading. A car previously selected for loading will have goods transferred from the warehouse to the car at the loading rate for teh commodity. The same routine is used for unloading, using the associated unloading rate.

Z018: Set Schedules to Active. Checks whether a schedule has been completed: if it has, then the schedule will be available for the next day. If the train is still running, then a new schedule will not be made available.

Z022: Refuel Locomotive. If spotted at a refuelling point, the locomotive will be refuleed and made available at the next MOPS hour.

Z023: Works Locomotives. As for Cars, but using the Locomotive Type to reset the Maintenance counter.

Alphabetical List of Commands

ABOUT	ABOUT MOPS
ACARXB	ALLOCATE CAR TO BLOCK
ACARXS	ALLOCATE CAR TO SET
ACARXT	ALLOCATE CAR TO TRAIN
ACTIVE	ACTIVATE SCHEDULE
ADAREA	ADD AREA
ADCART	ADD CAR TYPE
ADCARX	ADD CAR DETAIL
ADCLAS	ADD CAR CLASSIFICATION
ADCOMM	ADD COMMODITY
ADCURO	ADD CUSTOMER ROUTING INFORMATION
ADINDY	ADD INDUSTRY
ADINST	ADD INSTRUCTION
ADLOAD	ADD LOADING DEFINITIONS
ADLOCO	ADD LOCOMOTIVE
ADLOCT	ADD LOCOMOTIVE TYPE
ADPLAX	ADD PLACE
ADRAIL	ADD RAILROAD
ADROUT	ADD ROUTE
ADSCHD	ADD SCHEDULE
ADSECT	ADD ROUTE SECTION
ADSTAT	ADD STATION TYPE
ADSTAX	ADD STATION
ADTIMS	ADD SCHEDULE TIMINGS
ADUSER	ADD USER
ADWARE	ADD WAREHOUSE
ALOCOT	ALLOCATE LOCO TO TRAIN
ASSIST	LIST AVAILABLE COMMANDS
CARXAT	LOCATE CAR AT STATION
CARXSP	SPOT CAR
CHAREA	CHANGE AREA
CHCART	CHANGE CAR TYPE
CHCARX	CHANGE CAR DETAILS
CHCLAS	CHANGE CAR CLASSSIFICATION
CHCOMM	CHANGE COMMODITY
CHCURO	CHANGE CUSTOMER ROUTING INFORMATION
CHINDY	CHANGE INDUSTRY
CHLOAD	CHANGE LOADING DEFINITIONS
CHLOCO	CHANGE LOCOMOTIVE
CHLOCT	CHANGE LOCOMOTIVE TYPE
CHPARM	CHANGE PARAMETER
CHPASS	CHANGE PASSWORD
CHPLAX	CHANGE PLACE
CHRAIL	CHANGE RAILROAD
CHROUT	CHANGE ROUTE NAME
CHSCHD	CHANGE SCHEDULE
CHSTAT	CHANGE STATION TYPE
CHSTAX	CHANGE STATION
CHTIMS	CHANGE SCHEDULE TIMINGS
CHUSER	CHANGE USER
CHWARE	CHANGE WAREHOUSE DETAILS
CLEANX	SET CAR TO EMPTY/CLEAN
CPSCHD	COPY SCHEDULE
CPWARE	CHANGE PRODUCTION AT WAREHOUSE
CSPEED	SET MOPS CLOCK SPEED
DEMPTY	DETAIL EMPTY CAR REQUEST
DXAREA	DELETE AREA
DXCART	DELETE CAR TYPE

DXCARX	DELETE CAR
DXCLAS	DELETE CAR CLASSIFICATION
DXCOMM	DELETE COMMODITY
DXCURO	DELETE CUSTOMER ROUTING INFORMATION
DXINDY	DELETE INDUSTRY
DXINST	DELETE INSTRUCTION
DXLOAD	DELETE LOADING DEFINITION
DXLOCO	DELETE LOCOMOTIVE
DXLOCT	DELETE LOCOMOTVE TYPE
DXPLAX	DELETE PLACE
DXRAIL	DELETE RAILROAD
DXROUT	DELETE ROUTE
DXSCHD	DELETE SCHEDULE
DXSECT	DELETE ROUTE SECTION
DXSTAT	DELETE STATION TYPE
DXSTAX	DELETE STATION
DXUSER	DELETE USER
DXWARE	DELETE WAREHOUSE
EDUSER	ENABLE/DISABLE USER
ETRAIN	SET EXTRA TRAIN
FLASHX	FLASH MESSAGE
FUELXX	CHANGE LOCO FUEL STATE
HELP	GENERAL HELP
HOLIDX	SET HOLIDAY
LACARS	REPORT CARS BY STATUS
LDROUT	LIST DETAIL FOR SELECTED ROUTE
LDWARE	LIST WAREHOUSE DETAILS
LEMPTY	LIST EMPTY CAR REQUESTS
LIAREA	LIST AREAS
LICALX	LIST NEXT 10 DAYS
LICARS	LIST CARS
LICART	LIST CAR TYPES
LICLAS	LIST CAR CLASSIFICATIONS
LICOMM	LIST COMMODITIES
LICONS	REPORT CONSIST
LICURO	LIST CUSTOMER ROUTING INFORMATION
LIGEOG	PRINT GEOGRAPHY
LILOAD	LIST LOADING DEFINITIONS
LILOCO	LIST LOCOMOTIVES
LILOCT	LIST LOCOMOTIVE TYPES
LINEUP	REPORT LINE-UP
LIPARM	LIST PARAMETERS
LIPLAX	LIST PLACES
LIRAIL	LIST RAILROADS
LIROUT	LIST ALL ROUTES
LISCHD	LIST ALL SCHEDULES
LISTAT	LIST STATION TYPES
LISTAX	LIST STATIONS
LIUSER	LIST USERS
LIWARE	LIST WAREHOUSES
LMTCAR	REPORT UNALLOCATED EMPTY CARS
LOCOAT	LOCATE LOCO AT STATION
LOCOSP	SPOT LOCO
LONWAY	REPORT EMPTIES EN ROUTE
LORDER	LIST EMPTY AND WAYBILL REQUESTS
LSLOCO	LIST LOCOMOTIVE DETAILS
LSROUT	LIST SECTIONS FOR ROUTE
LSWARE	WAREHOUSES AT STATIONS
LTRAIN	START TRAIN AT LATER ORIGIN
LXAREA	SHOW AREAS FILE

LXCARS	SHOW CARS FILE
LXCART	SHOW CAR TYPES FILE
LXCLAS	SHOW CAR CLASSIFICATIONS FILE
LXCOMM	SHOW COMMODITIES FILE
LXCURO	SHOW CUSTOMER ROUTINGS FILE
LXLOAD	SHOW LOADING DEFINITIONS
LXLOCO	SHOW LOCOMOTIVE FILE
LXLOCT	SHOW LOCOMOTIVE TYPES
LXORDR	SHOW ORDERS FILE
LXPARM	SHOW PARAMETERS
LXPLAX	SHOW PLACES FILE
LXRAIL	SHOW RAILROAD FILE
LXROUT	SHOW ROUTES FILE
LXSCHD	SHOW SCHEDULES FILE
LXSECT	SHOW ALL SECTIONS
LXSTAT	SHOW STATION TYPE DATA
LXSTAX	SHOW STATIONS DATA
LXUSER	SHOW USER DATA
LXWARE	SHOW WAREHOUSE DATA
MAINTC	CHANGE CAR MAINTENANCE STATE
MAINTL	CHANGE LOCO MAINTENANCE STATE
MTYORD	ALLOCATE EMPTY TO ORDER
PDROUT	PRINT DETAIL FOR SELECTED ROUTE
PEMPTY	REPORT EMPTY CAR REQUESTS
POWERX	CHANGE LOCO POWER STATE
PRAREA	PRINT AREAS
PRCARS	PRINT CARS
PRCART	PRINT CAR TYPES
PRCLAS	PRINT CAR CLASSIFICATIONS
PRCOMM	PRINT COMMODITIES
PRCONS	PRINT CONSIST
PRCURO	PRINT CUSTOMER ROUTING INFORMATION
PRGEOG	PRINT GEOGRAPHY
PRLOAD	PRINT LOADING DEFINITIONS
PRLOCO	PRINT LOCOMOTIVES
PRLOCT	PRINT LOCOMOTIVE TYPES
PRPARM	REPORT PARAMETERS
PRPLAX	PRINT PLACES
PRRAIL	PRINT RAILROADS
PRROUT	PRINT ALL ROUTES
PRSCHD	PRINT ALL SCHEDULES
PRSTAT	PRINT STATION TYPES
PRSTAX	PRINT STATIONS
PRTIMS	PRINT TIMINGS FOR SELECTED SCHEDULE
PRUSER	PRINT USERS
PRWARE	PRINT WAREHOUSES
PSLOCO	PRINT LOCOMOTIVE DETAILS
PUBLSH	PUBLISH ROUTE
PXAREA	EXPORT AREAS
PXCARS	EXPORT CARS
PXCART	EXPORT CAR TYPES
PXCLAS	EXPORT CAR CLASSIFICATIONS
PXCOMM	EXPORT COMMODITIES
PXCURO	EXPORT ROUTINGS
PXLOAD	EXPORT LOADING DEFINITIONS
PXLOCO	EXPORT LOCOMOTIVES
PXLOCT	EXPORT LOCOMOTIVE TYPES
PXORDR	EXPORT ORDERS FILE
PXPARM	EXPORT PARAMETERS
PXPLAX	EXPORT PLACES

PXRAIL	EXPORT RAILROADS
PXROUT	EXPORT DETAIL FOR ALL ROUTES
PXSCHD	EXPORT ALL SCHEDULES
PXSECT	EXPORT SECTIONS FOR ALL ROUTES
PXSTAT	EXPORT STATION TYPES
PXSTAX	EXPORT STATIONS
PXTABL	EXPORT TIMETABLE
PXTIMS	EXPORT TIMINGS FOR ALL SCHEDULES
PXUSER	EXPORT USER DATA
PXWARE	EXPORT WAREHOUSES
REPORT	REPORT TRAIN
RESETP	RESET PASSWORD
SETTIM	SET MOPS DATE AND TIME
STRAIN	SET SCHEDULED TRAIN
TIMING	LIST TIMINGS FOR SELECTED SCHEDULE
TRAINS	LIST RUNNING TRAINS
TTRAIN	TERMINATE TRAIN
UNPUBL	SET PUBLISHED ROUTE TO DRAFT
UTRAIN	SET UNSCHEDULED TRAIN
VALIDR	VALIDATE ROUTE
XCARXB	REMOVE CAR FROM BLOCK
XCARXS	REMOVE CAR FROM SET
XCARXT	REMOVE CAR FROM TRAIN
XCTIVE	SET SCHEDULE INACTIVE
XLOCOT	REMOVE LOCO FROM TRAIN
XXSTOP	STOP MOPS