MOSEK Optimization Server

Release 8.1.0.63

MOSEK ApS

2018

CONTENTS

CHAPTER ONE

INTRODUCTION

The **MOSEK** Optimization Suite 8.1.0.63 is a powerful software package capable of solving large-scale optimization problems of the following kind:

- linear,
- conic quadratic (also known as second-order cone),
- convex quadratic,
- semidefinite,
- and general convex.

Integer constrained variables are supported for all problem classes except for semidefinite and general convex problems. In order to obtain an overview of features in the **MOSEK** Optimization Suite consult the product introduction guide.

The most widespread class of optimization problems is *linear optimization problems*, where all relations are linear. The tremendous success of both applications and theory of linear optimization can be ascribed to the following factors:

- The required data are simple, i.e. just matrices and vectors.
- Convexity is guaranteed since the problem is convex by construction.
- Linear functions are trivially differentiable.
- There exist very efficient algorithms and software for solving linear problems.
- Duality properties for linear optimization are nice and simple.

Even if the linear optimization model is only an approximation to the true problem at hand, the advantages of linear optimization may outweigh the disadvantages. In some cases, however, the problem formulation is inherently nonlinear and a linear approximation is either intractable or inadequate. *Conic optimization* has proved to be a very expressive and powerful way to introduce nonlinearities, while preserving all the nice properties of linear optimization listed above.

The fundamental expression in linear optimization is a linear expression of the form

$$Ax - b \in \mathcal{K}$$

where $\mathcal{K} = \{y : y \geq 0\}$, i.e.,

$$\begin{array}{l} Ax - b = y, \\ y \in \mathcal{K}. \end{array}$$

In conic optimization a wider class of convex sets $\mathcal{K}$ is allowed, for example in 3 dimensions $\mathcal{K}$ may correspond to an ice cream cone. The conic optimizer in **MOSEK** supports three structurally different types of cones $\mathcal{K}$, which allows a surprisingly large number of nonlinear relations to be modelled (as described in the **MOSEK** modeling cookbook), while preserving the nice algorithmic and theoretical properties of linear optimization.

1.1 Why the Optimization Server?

The **MOSEK** OptServer is a simple solver service. It can receive tasks over HTTP or HTTPS and return solutions, log and other information. It can be used either in

- *completely open mode*, where no authentication is required,
- *closed mode*, where authentication is required, or
- *semi-open mode*, where authentication is required for administrative tasks, but optimizer tasks can be submitted anonymously.

The OptServer provides an API for submitting tasks and retrieving information. It makes it easy to offload heavy computations to a remote machine. This is useful for running **MOSEK** on a wider range of devices.

CHAPTER TWO

CONTACT INFORMATION

Phone	+45 7174 9373	
Website	mosek.com	
Email		
	sales@mosek.com	Sales, pricing, and licensing
	support@mosek.com	Technical support, questions and bug reports
	info@mosek.com	Everything else.
Mailing Address		
	MOSEK ApS	
	Fruebjergvej 3	
	Symbion Science Park, Box 16	
	2100 Copenhagen O	
	Denmark	

You can get in touch with **MOSEK** using popular social media as well:

Blogger	http://blog.mosek.com/
Google Group	https://groups.google.com/forum/#!forum/mosek
Twitter	https://twitter.com/mosektw
Google+	https://plus.google.com/+Mosek/posts
Linkedin	https://www.linkedin.com/company/mosek-aps

In particular **Twitter** is used for news, updates and release announcements.

CHAPTER

THREE

LICENSE AGREEMENT

Before using the **MOSEK** software, please read the license agreement available in the distribution at `<MSKHOME>/mosek/8/mosek-eula.pdf` or on the **MOSEK** website https://mosek.com/products/license-agreement.

MOSEK uses some third-party open-source libraries. Their license details follows.

zlib

MOSEK includes the *zlib* library obtained from the zlib website. The license agreement for *zlib* is shown in Listing 3.1.

Listing 3.1: *zlib* license.

```
zlib.h -- interface of the 'zlib' general purpose compression library
version 1.2.7, May 2nd, 2012

Copyright (C) 1995-2012 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied
warranty.  In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly        Mark Adler
jloup@gzip.org          madler@alumni.caltech.edu
```

fplib

MOSEK includes the floating point formatting library developed by David M. Gay obtained from the netlib website. The license agreement for *fplib* is shown in Listing 3.2.

Listing 3.2: *fplib* license.

```
/****************************************************************
 *
```

CHAPTER

FOUR

INSTALLATION

4.1 What's in the box

The **MOSEK** OptServer is shipped as part of the **MOSEK** Optimization Suite. It is located in the folder `MSKHOME/mosek/8/opt-server`, where `MSKHOME` is the folder **MOSEK** has been installed in.

The OptServer files are organized in the following folders:

- `bin` – It contains the executables. i.e.
  - `MosekServer`, see Sec. 4.4.
  - `install_MosekServer`, see Sec. 4.1.
- `var` – It contains the working directories and the GUI elements:
- `etc` – It contains default configuration file and security key.

4.2 Requirements

OptServer only requires Python 3+ to run the configuration script. It has been tested only on Linux 64 bit.

4.3 Installation

To install the OptServer you need to execute the `install_MosekServer` located in the `MSKDIR/opt-server/` folder. A set of options can be provided to customize the installation:

`--inplace`
: Set up the server to run directly in the unpacked distro directory

`--user`
: Install in `~/.local`

`--global`
: Install in /usr

`--prefix DIR`
: Install in DIR

`--certdir DIR`
: This directory contains valid `cert.pem` and `key.pem` for `HTTPS`. If `DIR=''`, then `HTTPS` is disabled.

`--password PWD`
: Password for initial user *admin*

`--port PORT`
: Configure server to listen to port `PORT`

`--disable-gui`
: Disable GUI and API

`--enable-get`
: Enable fetching submitted data/problem files

`--enable-anonymous`
: Enable submitting without credentials

If the installation succeeds, you can then run the OptServer as described in Sec. 4.4.

4.4 Running the Server

The OptServer can be started by running the executable `MosekServer` from the `OPT_SERVER_HOME/bin` folder, for instance

```
$ $OPT_SERVER_HOME/bin/MosekServer
```

With no command line the server runs using the configurations setup during the installation process, see Sec. 4.1. To override the configuration set in the installation, several options can be passed to the server.

Note: Options can be prefixed by a single or a dobule dash, i.e. either `-` or `--`.

Some examples follow.

Switching debug mode on

If the server is not working a expected, it may be useful to turn on debugging:

```
MosekServer -debug=true
```

Change the port

Changing the port is a pretty standard step.

```
MosekServer --port=30080
```

4.5 OptServer Options

The complete list of options follow.

`-base={/var/Mosek/server}`
: Base directory

`-certdir=/etc/Mosek/server/cert`
: Enable SSL, `cert.pem` and `key.pem` in this directory

`-cmd=$basedir/script/solve.py $workdir $task`
: Solver command

`-config=/etc/Mosek/server.conf`
: Specify configuration file (JSON)

`-debug=false`
: Turn on debugging info (turned off by default). This is a boolean option.

`-enable-anonymous-submit=false`
: Enable anonymous submitting (turned off by default).

`-enable-get-problem=false`
: Enable fetching submitted tasks (turned off by default).

`-enable-user-api=false`
: Enable extended programming API

`-enable-user-gui=false`
: Enable user interface (turned off by default).

`-hostname=hostname`
: Server host name

`-logfile=filename`
: Log file name

`-login-expiry=86400`
: Login expiry time in seconds

`-port=30080`
: Port to listen to

`-staticdir=extern`
: Directory with files served under `/static/`

CHAPTER

FIVE

OVERVIEW

In this section an overview of the basic concepts about the OptServer is given.

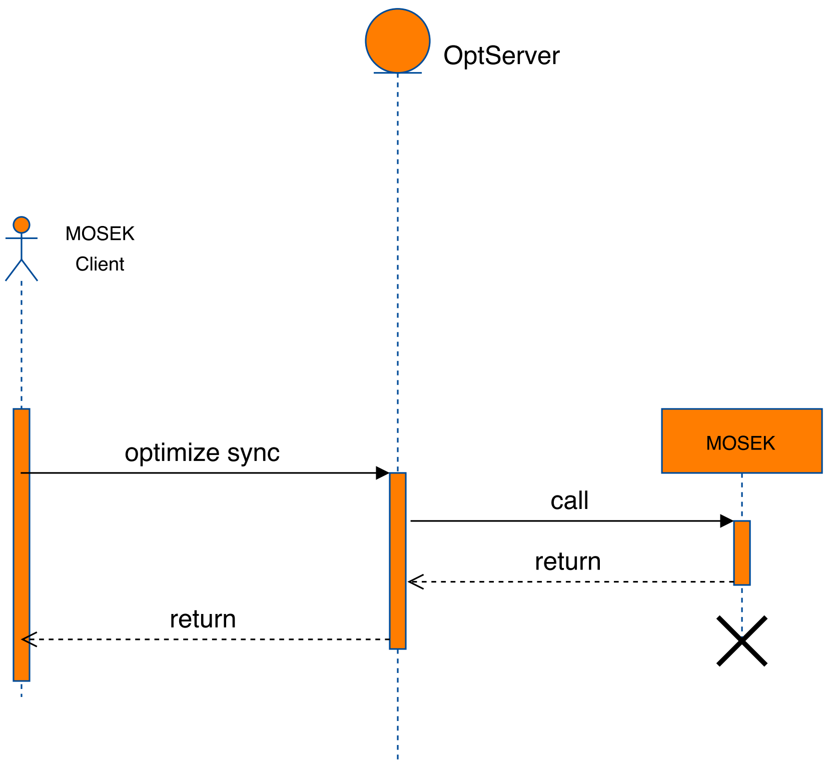
- *Synchronous Job submission*
- *Asynchronous Job submission*

5.1 Synchronous Optimization

The easiest way to submit optimization problem to the OptServer is in *synchronous mode*:

1. A submision request is sent over the OptServer and the problem is transferred.
2. The submitter is put on hold.
3. The OptServer runs the optimizer and wait for the results.
4. When the optimizer terminates the OptServer collects the outcome and passes over the client.
5. The client receives the solution and get back control.

The process can be represented as in Fig. 5.1.



The workflow is simple and effective for problems that does not take long to solve, or at least in all settigns in which the client can wait for the job to complete.

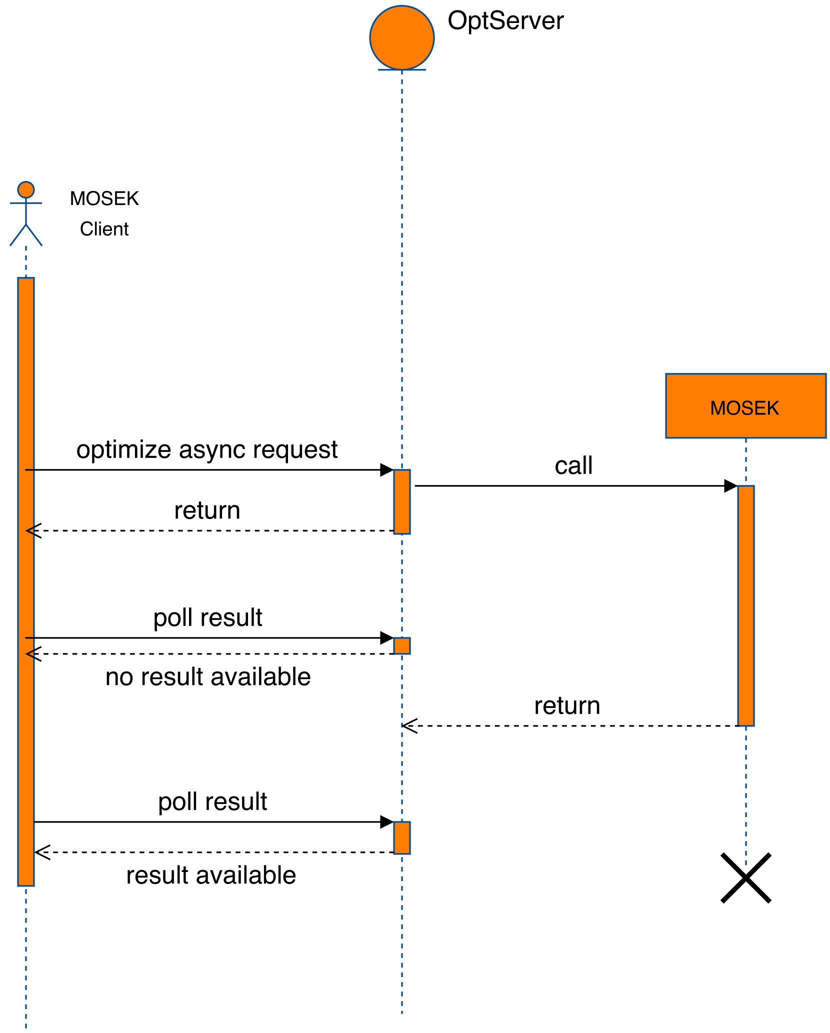
Warning: If the connection between the client and the OptServer is lost, the job result can only be recovered by manually accesing the job workspace on the server.

5.2 Asynchronous Optimization

The OptServer accepts jobs in *asynchronous mode*, i.e. once the job request is accepted, the client get control back and the server proceed with the optimization. The client can query the OptServer for the status of the job and in case retrieve the solution, if any. The steps can be summarized as:

1. A submision request is sent over the OptServer and the problem is transferred.
2. The submitter regain control and can continue its execution flow.
3. The client can query the OptServer at any time about the job status and solution availability.
4. The OptServer runs the optimizer and wait for the results.
5. When the optimizer terminates the OptServer collects the outcome.

The process can be represented as in Fig. 5.2.



Asynchronous mode is particularly suitable when

- a job is expected to run for long time,
- one must submit a set of small jobs that can run in parallel or
- the submitter is a short-lived process, such as a *docker instance*.

Warning: The OptServer does not implement any queing strategy, so a job is executed as it is submitted.

CHAPTER

SIX

GUIDELINES

6.1 Known Limitations

The main limitation in the use of the **MOSEK** Optimization Server 8.1.0.63 are reported in this section.

Platforms

Currently the **MOSEK** OptServer has been only tested on Linux 64bit machines.

Compatibility

Job submission using **MOSEK** API is only available starting from **MOSEK** 8.

Submitting jobs directly using HTTP commands is possible as long as the file format is accepted. However, it must be noticed that the compatibility does not depend on the OptServer but on the underlying **MOSEK** solver available on server machine.

6.2 Resources and performance

OptServer is a very lightweight server and requires very limited resources both in terms of memory and CPU.

CPU/Memory

The use of CPU/memory resources by OptServer should be neglegible.

Network

Most of the network load is due to the transfer of the optimization problem from the client to the server. That happens in a single burnst. Therefore

- For long running jobs the transfer time is typically neglegible,
- For easy to solve problems the transfer time may be more significant.

Hint: The same problem can result in file of different sizes depending on the choosen format.

However, OptServer has not been designed for time critical production environment.

Disk usage

Each job is stored on disk along with log and solutions. The reasons are:

1. avoid to keep jobs in memory while the solver is running,
2. in case of crash, information can be recovered from the disk,
3. solution and result can be recovered asynchronously reading from disk.

Therefore a suitable amount of free space must be available.

The folder used to store jobs information is under the *basedir* folder, that can be set in the configuration file.

Note: OptServer does not delete data for completed jobs. Users of cloud services should take some care in case they pay storage fees.

CHAPTER

SEVEN

TUTORIALS

This section contains tutorials that illustrate how communicate with the **MOSEK** OptServer in order to

- offload optimization problem from the client to the server and
- retrieve the solution and the solver log.

The tutorials are implemented using the Python *3* programming language. The reason for this choice is that Python provides an easy-to-use HTTP client and allows for a simple and compact code.

- *Submission and solution in synchronous mode*
- *Submission and solution in asynchronous mode*
- *Submission and solution using Condor*

7.1 Synchronous Problem Submission

This tutorial shows how to

- submit a job to the OptServer,
- remotely run **MOSEK** and wait for the optimization to terminate and
- retrieve the solution and log.

The optimization problem is assumed to be stored in a file using one of the available *file formats*.

The connection is managed using the `http` Python module, and it is assumed to be established successfully: as a result, an object `con` is available to manage the connection.

First of all, the problem is submitted to the OptServer by a *submit* command

Listing 7.1: How to submit a job to the OptServer.

```
con.request("POST",'/submit', dataf)
resp = con.getresponse()
```

If no errors have occourred, a request for running the optimizer can be sent

Listing 7.2: How to run an optimization job with OptServer.

```
con.request("GET","/solve?token="+token)
resp = con.getresponse()
```

At this point the request wil return when the optimization terminates. If no errors have happened, the results are already available

Listing 7.3: How to get the results.

```
            res = resp.getheader('X-Mosek-Res-Code',None)
            trm = resp.getheader('X-Mosek-Trm-Code',None)

            print("\tMOSEK response: %s" % res)
            print("\t      trm resp: %s" % trm)
            if resp.status == http.client.OK:
                print("Solution:")
                print(resp.read().decode('ascii',errors='ignore'))
```

The log is readly available as well

Listing 7.4: How to retrieve the log for a job on OptServer.

```
            con.request("GET","/log?token="+token)
            resp = con.getresponse()
```

The whole example is in Listing 7.5.

Listing 7.5: How to submit a job and solve the problem synchronously.

```
import http.client
import sys


def check_status(resp):

    print("\tHTTPResponse: %s / %s" % (resp.status,resp.reason))
    for k,v in  resp.getheaders():
        print("\t%s: %s" % (k,v))

    if resp.status not in [http.client.OK, http.client.NO_CONTENT]:
        print("An error occourred!")
        sys.exit(1)


if __name__ == '__main__':
    host = sys.argv[1]
    port = int(sys.argv[2])
    probfile = sys.argv[3]


    con = http.client.HTTPConnection(host,port)
    try:
        with open(probfile,'rb') as dataf:
            ## Submit job
            print("POST /submit")

            con.request("POST",'/submit', dataf)
            resp = con.getresponse()
            check_status(resp)

            token = resp.read().decode('ascii')

            ## Solve and wait for solution
            print("GET /solve")
            con.request("GET","/solve?token="+token)
            resp = con.getresponse()
            check_status(resp)
            res = resp.getheader('X-Mosek-Res-Code',None)
            trm = resp.getheader('X-Mosek-Trm-Code',None)
```

```
            print("\tMOSEK response: %s" % res)
            print("\t      trm resp: %s" % trm)
            if resp.status == http.client.OK:
                print("Solution:")
                print(resp.read().decode('ascii',errors='ignore'))
            print("GET /log")

            con.request("GET","/log?token="+token)
            resp = con.getresponse()

            check_status(resp)

            if resp.status == http.client.OK:
                print(resp.read().decode('utf-8',errors='ignore'))
    finally:
        con.close()
```

7.2 Asynchronous Problem Submission

This tutorial shows how to

- submit a job to the OptServer,
- start the optimization job running **MOSEK** on server side,
- closing the connection to the server and
- connect again and retrieve te solution and log.

The optimization problem is assumed to be stored in a file using one of the available *file formats*. The connection is managed using `http` Python module, and it is assumed to be establish successfully.

First of all, the problem is submitted to the OptServer by a `POST` operation

Listing 7.6: How to submit a job to the OptServer.

```
            con.request("POST",'/submit', dataf)
            resp = con.getresponse()
```

Note that this operation is identical to the *synchronuous case*.

If no errors have occourred, a request for running the optimizer can be sent

Listing 7.7: How to run an optimization job with OptServer.

```
            con.request("GET","/solve-background?token="+token)
            resp = con.getresponse()
```

The program regains control immediately. The connection is then closed and reopened, to make sure client and server are working asynchronously.

If no errors have happened, the results can be retrieved

Listing 7.8: How to get the results.

```
            con.request("GET","/solution?token="+token)
            resp = con.getresponse()

            check_status(resp)

            res = resp.getheader('X-Mosek-Res-Code',None)
            trm = resp.getheader('X-Mosek-Trm-Code',None)
```

```
            print("\tMOSEK response: %s" % res)
            print("\t       trm resp: %s" % trm)
            if resp.status == http.client.OK:
                print("Solution:")
                print(resp.read().decode('ascii',errors='ignore'))
```

The log is readily available as well

Listing 7.9: How to retrieve the log for a job on OptServer.

```
            con.request("GET","/log?token="+token)
            resp = con.getresponse()

            if resp.status == http.client.OK:
                print(resp.read().decode('utf-8',errors='ignore'))
```

The whole example is in Listing 7.10.

Listing 7.10: How to submit a job and solve the problem asynchronously.

```
import http.client
import sys

def check_status(resp):

    print("\tHTTPResponse: %s / %s" % (resp.status,resp.reason))
    for k,v in  resp.getheaders():
        print("\t%s: %s" % (k,v))

    if resp.status not in [http.client.OK, http.client.NO_CONTENT]:
        print("An error occourred!")
        sys.exit(1)

if __name__ == '__main__':
    host = sys.argv[1]
    port = int(sys.argv[2])

    probfile = sys.argv[3]

    token=[]
    con = http.client.HTTPConnection(host,port)
    try:
        with open(probfile,'rb') as dataf:
            ## Submit job
            print("POST /submit")
            con.request("POST",'/submit', dataf)
            resp = con.getresponse()
            check_status(resp)

            token = resp.read().decode('ascii')

            ## Start solving end close connection
            print("GET /solve-background")
            con.request("GET","/solve-background?token="+token)
            resp = con.getresponse()
            check_status(resp)

    finally:
        con.close()
        print("connection closed")


    con = http.client.HTTPConnection(host,port)
```

```
    print("connection open")

    try:
            print("GET /solution")
            con.request("GET","/solution?token="+token)
            resp = con.getresponse()

            check_status(resp)

            res = resp.getheader('X-Mosek-Res-Code',None)
            trm = resp.getheader('X-Mosek-Trm-Code',None)

            print("\tMOSEK response: %s" % res)
            print("\t      trm resp: %s" % trm)
            if resp.status == http.client.OK:
                print("Solution:")
                print(resp.read().decode('ascii',errors='ignore'))

            print("GET /log")
            con.request("GET","/log?token="+token)
            resp = con.getresponse()

            if resp.status == http.client.OK:
                print(resp.read().decode('utf-8',errors='ignore'))
    finally:
        con.close()
```

7.3 Problem Submission via Condor

HTCondor, formerly known as *Condor* is

> "a specialized workload management system for compute-intensive jobs."

This tutorial shows how to submit optimization problem to a *HTCondor* server via OptServer.

The idea is very simple: since OptServer executes **MOSEK** using a simple Python script (`solve.py`), we can instruct OptServer to use a different script that will interface with *HTCondor*. To this extent we use the script as in Listing 7.11.

Listing 7.11: An example of script to off-load a job from OptServer to a *HTCondor* server.

```
import sys
import os,os.path
import subprocess

if __name__ == '__main__':
  workdir  = sys.argv[1]
  probfile = sys.argv[2]
  pidfile = os.path.join(workdir,"PID")

  with open(pidfile,'wt', encoding='ascii')  as f:
     f.write(str(os.getpid()))

  r = 1
  try:
      r = subprocess.call(['condor_run',
                           os.path.abspath(os.path.join(os.path.dirname(__file__),"solve.py")),
                           workdir,
                           probfile,
                           '-noPID'])
  finally:
```

```
21     try:
22         os.remove(pidfile)
23    except:
24         pass
25
26 sys.exit(r)
```

The script operates as follows:

- lines `10-11`: the job PID is stored in a text file called `PID` in the working directory;
- lines `14-24` : a *HTCondor* process is created, responsible to run the `solve.py` script.

To tell OptServer to use the script in Listing 7.11 instead of the default `solve.py`, the `cmd` option (see Sec. 9) in the configuration file `server.conf` must be modified accordingly. In this case the script is available in the `script` directory of the OptServer distribution. Therefore the configuration file can be simply modified changing the `cmd` option to

```
"cmd"   : "${CONFIGDIR}/script/tocondor.py ${TASK}",
```

CHAPTER

EIGHT

SECURITY

OptServer uses on HTTPS by default, for which a self-signed demo certificate is provided in `security/cert`. The user can point OptServer to another certificate setting the `certdir` option (see Sec. 9).

8.1 User role management

Users can be *registered* or *anonymous*. A registered user can be either

- *administrator* or
- *submitter* or
- *anonymous*.

Users can be added, removed or their status changed from the web interface (see Sec. 10).

Administrator

An administrator can submit jobs and perform all administrative tasks.

It is also possible to grant temporary *administrator* access. Any *administrator* can log in the web interface and grant a access token to a user, from the *tokens* page. Each temporary token is specified in terms of how long it lasts.

Submitter

This users can both submit jobs and access the web interface to

- collect information about their own jobs,
- modify their own information but
- they can not perfomr administrative task.

Anonymous

Anonymous users are not allowed by default, unless the `enable-anonymous-submit` is specified. Anonymous users can only submit jobs.

CHAPTER NINE

CONFIGURATION

The configuration of the OptServer is stored in a single flat JSON file. Following keys are recognized:

`address <string>`
: Host and port, in the format `HOST:PORT`. If `HOST` is left blank then `localhost` is used

`basedir <string>`
: Work directory.

`certdir <string>`
: Directory containing `key.pem` and `cert.pem`.

`externdir <string>`
: Directory containing passive files (css, javascript, images etc.) that is required by the web pages

`enable-login <true|false>`
: Enable login and management.

`enable-management <true|false>`
: Enable management, even when login is disabled. Forced to `true` if *enable-login* is `true`

`enable-anonymous-submit <true|false>`
: Allow submitting tasks without authentication.

`login-expiry <integer>`
: Expiry of login session in seconds.

`password-salt <string>`
: Name of the file used for password salting.

`cmd <string>`
: The command executed to solve problems.

  The `cmd` key allows for variable substitution using `${...}`. Following variables are recognized:

  - *BASEDIR*
  - *CONFIGDIR*
  - *TASK* Name of the problem file.
  - *WORKDIR* Name of the working directory for the task.

If a key is not specified, then its default value, if any, is used.

The default configuration is stored in the `server.conf` file and reported in Listing 9.1.

Listing 9.1: The OptServer default configuration.

```
{
 "address"        : ":30080",
 "basedir"        : "run",
 "externdir"      : "../management/extern",
 "logfile"        : "run/server.log",
 "pidfile"        : "run/PID",
 "cmd"            : "${CONFIGDIR}/script/solve.py ${TASK}",
```

```
 "enable-login"  : true,
 "certdir"       : "security/cert",
 "password-salt" : "./run/salt"
}
```

CHAPTER

TEN

WEB GUI INTERFACE

The **MOSEK** OptServer provides a minimalistic web interface that allows to

- monitor and terminate jobs and
- grant or revoke access tokens,

The web interface can be activated setting the options `enable-management` or `enable-login` (see Sec. 9). By default is not active.

10.1 Login page

When the user opens the web interface a login page is shown. The user must input its user name and in order to be authenticated. See Sec. 8 for further details.

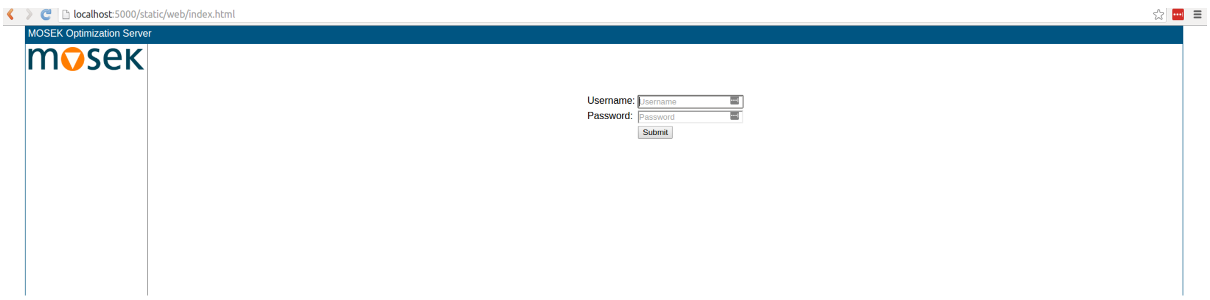


Fig. 10.1: A screenshot of the login page.

Note: The OptServer does not keep trace of login attempts.

If the login is successful the user is presented with a minimal login page.

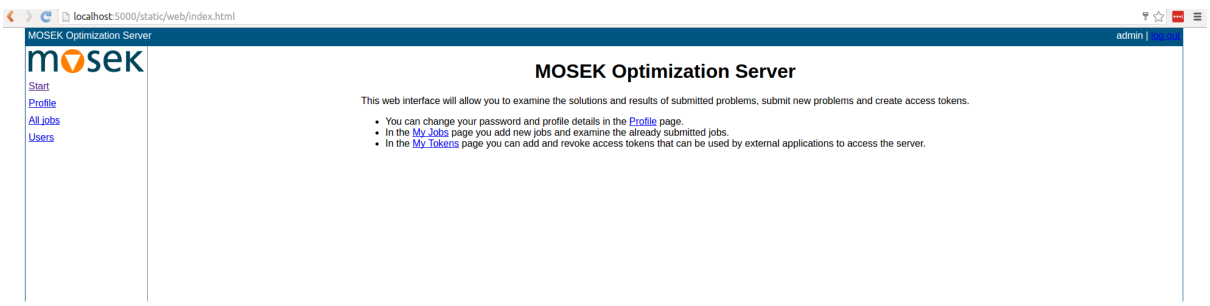


Fig. 10.2: A screenshot of the main page.

10.2 Job list page

Selecting the `All Jobs` or `My Jobs` link from the left sidebar for administrators and submitters respectively, the job list page is visualize. It provides information about jobs and for submitters the possibility to post a new job.

An example of how the page looks like is in Fig. 10.3.

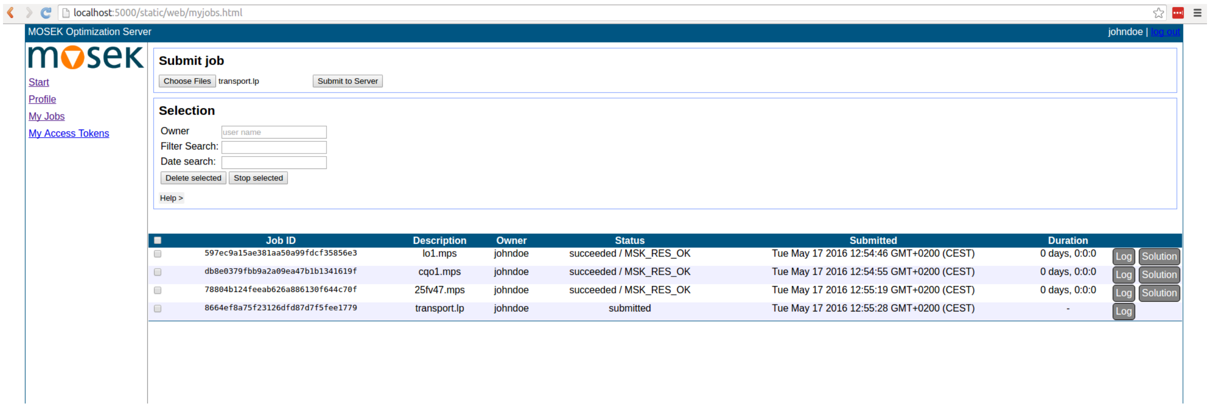


Fig. 10.3: A screenshot of the job list page.

10.2.1 The Job List

The list contains all jobs that have been submitted by the user if it a submitter, or all users for administrators. For each job the following information are available

- **Job ID**: a unique job id,
- **Description**: the submitted file name,
- **Owner**: the user that submit the job (left blank if anonymous),
- **Status**: whether the job is running, complete with success or something went wrong,
- **Submitted**: when the job has been submitted,
- **Duration**: the overall duration of the job.

For each job two buttons are available that will provide additional information directly from the solver execution:

- **Log**: shows the solver log for that job,
- **Solution**: the solution for the job.

Note: **Log** and **Solution** are displayed in a new window.

10.2.2 Job List Filter

To navigate among the submitted jobs OptServer provides a simple search tool that combines criteria from two fields:

- *Filter Search* : Entries in the text search box are matched against values in the columns “Job ID”, “Descripion”, “Owner” and “Status”. If the string is a substring in any of those fields the row is a match.

- *Date Search* : The entry should either be a single data or a date range ("start .. end"). A date is written as

  ```
  December 21 2012 12:45:00
  2012-12-21 12:45:00
  ```

  and the various parts can be left out, i.e. the following are valid

  ```
  Dec 21 2012 meaning Dec 21 2012 00:00:00
  2012 meaning Jan 1 2012 00:00:00
  ```

  Ranges are specified with the `..`, and either end can be left blank to indicate no bound, e.g.

  ```
  Dec 21 2012 .. meaning Dec 21 2012 00:00:00 until now
  .. -2012 meaning anything before Jan 1 2012 00:00:00
  2011..2012 meaning anything bwteen Jan 1 2011 and Jan 1 2012
  ```

Some examples follow

All jobs submitted in May 2016

In the *Date Search* field enter

```
May .. 2016
```

All jobs submitted by users with name `john`

In the *Filter Text* field input

```
john
```

All failed jobs

In the *Filter Text* field input

```
failed
```

10.2.3 Job submission

If the user is a *submitter* it can directly submit a job in asynchronous mode using the GUI.

10.2.4 Job Status

The possible job statuses are listed in Table 10.1.

Table 10.1: Status keys.

Status key	Description
submitted	The job has just been submitted, waiting to run.
running	The job is currently running.
failed	The solver did not terminated correctly.
done	The solver terminated correctly with a response code.

Details follow.

Status *submitted*

The *submitted* status indicates that the job has been received and stored. It is in the process to be executed.

Note: OptServer does not provide any queing system. Therefore a job is never waiting for execution.

Status *running*

A job is *running* when the solver has been started but not yet terminated.

Status *failed*

The *failed* status indicates that something wrong has happened. Two scenarios apply:

1. The running script returned an error before the solvr could start.
2. The solver did started, but it terminated unexpectedly and providing no error code. This is the situation for instance in which a serious bug leads to a *segmentation fault*. The *log* may provide useful information on the reason the crash happend.

Tip: Please consider making a bug report whenever a job fails.

Status *done*

Whenever the job terminates in a nice and controlled way it is flagged as *done*. This *does not* imples the optimization has been successful, but only that the solver has terminated its execution and returned a response code. To distinguish among the different scenarios OptServer also provides the solver response code, which clearly informs the user how and why the solver stopped.

For example

- `MSK_RES_OK` indicates the solver terminate successfully,
- `MSK_RES_ERR_LICENSE_EXPIRED` indicates the **MOSEK** license has expired,
- `MSK_RES_TRM_MAX_TIME` indicates the solver termianted because the maximum allowed time has reached.

The log contains more detailed information.

10.3 User Page

In this page OptServer list its users, organized in a sortable table that report

- the username
- the full name,
- the email and
- the roles.

To get more information about the roles see Sec. 8.

In this page administrators can:

- list the OptServer users,
- delete/create users and
- update user information.

An example of how the page looks like is in Fig. 10.4.

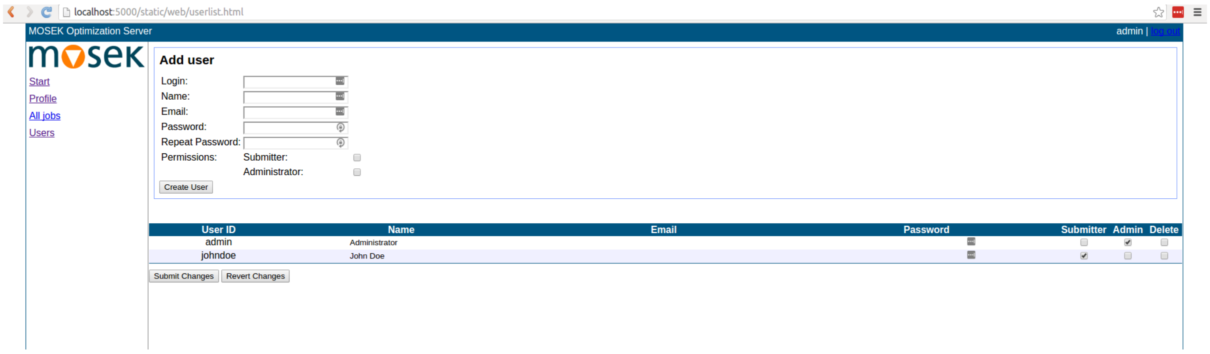


Fig. 10.4: A screenshot of the user page.

Add New User

A new user can be add using the form on top of the page. Compulsory fields are

- **Login**,
- **Password**.

A user is by default neither an administrator nor a submitter.

Modify an Existing User

Administrators can modify the user information by editing the relevant fields directly in the user list: just click with mouse pointer and edit!

Delete an Existing User

An existing user can be delete only by an administrator. This operation will not completely wipe out the user from the OptServer:

- the job submitted by the user will still be listed,
- all temporary files stored (problems and their log and solutions) will not be deleted.

Warning: If a new user with the same login name is created, it will take over all date from the deleted user!

CHAPTER

ELEVEN

OPTSERVER REFERENCE

11.1 OptServer protocol

The server protocol is HTTP plus a couple of extension headers. Commands that the client can use:

`POST /api/submit`

Post a problem to the server.

On response `OK`, a token identifying the problem is returned in the response body, and the token cookie is set to the token string. If logins are disabled or anonymous submits are allowed, no authentication is required. Otherwise, one of following is required:

- An access token, passed in the query part as `access-token=...`, or
- a valid session, passed in the session cookie (i.e. a user that is logged in can submit).

`GET /api/solve`

Start solving and wait for the solver to finish.

The token is passed either as a query string `token=<tokenstr>`, or in the token cookie. The Accept header identifies the accepted solution formats. Currently recognized formats are:

`application/x-mosek-task`	Request solution in `.task` format.
`application/x-mosek-json`	Request solution in JSON format.
`text/plain`	Request a plain ASCII formatted solution

`GET /api/solve-background`

Start solving the identified task in background, return immediately.

It returns `OK` is the solver started sucessfully. The token is passed either as a query string `token=<tokenstr>`, or in the token cookie.

`GET /api/log`

Return the log.

The token is passed either as a query string `token=<tokenstr>`, or in the token cookie. If the query string contains the parameter `offset=XXXX`, the log file will be returned from offset *XXXX*.

`GET /api/solution`

Return the solution

It returns the solution if available, or `NO_CONTENT` if the tasks exists but no solution is available. The token is passed either as a query string `token=<tokenstr>`, or in the token cookie.

`HEAD /api/break`

Attempt to terminate the solver.

The token is passed either as a query string `token=<tokenstr>`, or in the token cookie.

11.2 Parameters grouped by topic

Analysis

- *MSK_DPAR_ANA_SOL_INFEAS_TOL*
- *MSK_IPAR_ANA_SOL_BASIS*
- *MSK_IPAR_ANA_SOL_PRINT_VIOLATED*
- *MSK_IPAR_LOG_ANA_PRO*

Basis identification

- *MSK_DPAR_SIM_LU_TOL_REL_PIV*
- *MSK_IPAR_BI_CLEAN_OPTIMIZER*
- *MSK_IPAR_BI_IGNORE_MAX_ITER*
- *MSK_IPAR_BI_IGNORE_NUM_ERROR*
- *MSK_IPAR_BI_MAX_ITERATIONS*
- *MSK_IPAR_INTPNT_BASIS*
- *MSK_IPAR_LOG_BI*
- *MSK_IPAR_LOG_BI_FREQ*

Conic interior-point method

- *MSK_DPAR_INTPNT_CO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_CO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_REL_GAP*

Data check

- *MSK_DPAR_DATA_SYM_MAT_TOL*
- *MSK_DPAR_DATA_SYM_MAT_TOL_HUGE*
- *MSK_DPAR_DATA_SYM_MAT_TOL_LARGE*
- *MSK_DPAR_DATA_TOL_AIJ*
- *MSK_DPAR_DATA_TOL_AIJ_HUGE*
- *MSK_DPAR_DATA_TOL_AIJ_LARGE*
- *MSK_DPAR_DATA_TOL_BOUND_INF*
- *MSK_DPAR_DATA_TOL_BOUND_WRN*
- *MSK_DPAR_DATA_TOL_C_HUGE*
- *MSK_DPAR_DATA_TOL_CJ_LARGE*
- *MSK_DPAR_DATA_TOL_QIJ*
- *MSK_DPAR_DATA_TOL_X*
- *MSK_DPAR_SEMIDEFINITE_TOL_APPROX*
- *MSK_IPAR_CHECK_CONVEXITY*
- *MSK_IPAR_LOG_CHECK_CONVEXITY*

Data input/output

- *MSK_IPAR_INFEAS_REPORT_AUTO*
- *MSK_IPAR_LOG_FILE*
- *MSK_IPAR_OPF_MAX_TERMS_PER_LINE*
- *MSK_IPAR_OPF_WRITE_HEADER*
- *MSK_IPAR_OPF_WRITE_HINTS*
- *MSK_IPAR_OPF_WRITE_PARAMETERS*
- *MSK_IPAR_OPF_WRITE_PROBLEM*
- *MSK_IPAR_OPF_WRITE_SOL_BAS*
- *MSK_IPAR_OPF_WRITE_SOL_ITG*
- *MSK_IPAR_OPF_WRITE_SOL_ITR*
- *MSK_IPAR_OPF_WRITE_SOLUTIONS*
- *MSK_IPAR_PARAM_READ_CASE_NAME*
- *MSK_IPAR_PARAM_READ_IGN_ERROR*
- *MSK_IPAR_READ_DATA_COMPRESSED*
- *MSK_IPAR_READ_DATA_FORMAT*
- *MSK_IPAR_READ_DEBUG*
- *MSK_IPAR_READ_KEEP_FREE_CON*
- *MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU*
- *MSK_IPAR_READ_LP_QUOTED_NAMES*
- *MSK_IPAR_READ_MPS_FORMAT*

- *MSK_IPAR_READ_MPS_WIDTH*
- *MSK_IPAR_READ_TASK_IGNORE_PARAM*
- *MSK_IPAR_SOL_READ_NAME_WIDTH*
- *MSK_IPAR_SOL_READ_WIDTH*
- *MSK_IPAR_WRITE_BAS_CONSTRAINTS*
- *MSK_IPAR_WRITE_BAS_HEAD*
- *MSK_IPAR_WRITE_BAS_VARIABLES*
- *MSK_IPAR_WRITE_DATA_COMPRESSED*
- *MSK_IPAR_WRITE_DATA_FORMAT*
- *MSK_IPAR_WRITE_DATA_PARAM*
- *MSK_IPAR_WRITE_FREE_CON*
- *MSK_IPAR_WRITE_GENERIC_NAMES*
- *MSK_IPAR_WRITE_GENERIC_NAMES_IO*
- *MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS*
- *MSK_IPAR_WRITE_INT_CONSTRAINTS*
- *MSK_IPAR_WRITE_INT_HEAD*
- *MSK_IPAR_WRITE_INT_VARIABLES*
- *MSK_IPAR_WRITE_LP_FULL_OBJ*
- *MSK_IPAR_WRITE_LP_LINE_WIDTH*
- *MSK_IPAR_WRITE_LP_QUOTED_NAMES*
- *MSK_IPAR_WRITE_LP_STRICT_FORMAT*
- *MSK_IPAR_WRITE_LP_TERMS_PER_LINE*
- *MSK_IPAR_WRITE_MPS_FORMAT*
- *MSK_IPAR_WRITE_MPS_INT*
- *MSK_IPAR_WRITE_PRECISION*
- *MSK_IPAR_WRITE_SOL_BARVARIABLES*
- *MSK_IPAR_WRITE_SOL_CONSTRAINTS*
- *MSK_IPAR_WRITE_SOL_HEAD*
- *MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES*
- *MSK_IPAR_WRITE_SOL_VARIABLES*
- *MSK_IPAR_WRITE_TASK_INC_SOL*
- *MSK_IPAR_WRITE_XML_MODE*
- *MSK_SPAR_BAS_SOL_FILE_NAME*
- *MSK_SPAR_DATA_FILE_NAME*
- *MSK_SPAR_DEBUG_FILE_NAME*
- *MSK_SPAR_INT_SOL_FILE_NAME*
- *MSK_SPAR_ITR_SOL_FILE_NAME*
- *MSK_SPAR_MIO_DEBUG_STRING*
- *MSK_SPAR_PARAM_COMMENT_SIGN*

- *MSK_SPAR_PARAM_READ_FILE_NAME*
- *MSK_SPAR_PARAM_WRITE_FILE_NAME*
- *MSK_SPAR_READ_MPS_BOU_NAME*
- *MSK_SPAR_READ_MPS_OBJ_NAME*
- *MSK_SPAR_READ_MPS_RAN_NAME*
- *MSK_SPAR_READ_MPS_RHS_NAME*
- *MSK_SPAR_SENSITIVITY_FILE_NAME*
- *MSK_SPAR_SENSITIVITY_RES_FILE_NAME*
- *MSK_SPAR_SOL_FILTER_XC_LOW*
- *MSK_SPAR_SOL_FILTER_XC_UPR*
- *MSK_SPAR_SOL_FILTER_XX_LOW*
- *MSK_SPAR_SOL_FILTER_XX_UPR*
- *MSK_SPAR_STAT_FILE_NAME*
- *MSK_SPAR_STAT_KEY*
- *MSK_SPAR_STAT_NAME*
- *MSK_SPAR_WRITE_LP_GEN_VAR_NAME*

Debugging

- *MSK_IPAR_AUTO_SORT_A_BEFORE_OPT*

Dual simplex

- *MSK_IPAR_SIM_DUAL_CRASH*
- *MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION*
- *MSK_IPAR_SIM_DUAL_SELECTION*

Infeasibility report

- *MSK_IPAR_INFEAS_GENERIC_NAMES*
- *MSK_IPAR_INFEAS_REPORT_LEVEL*
- *MSK_IPAR_LOG_INFEAS_ANA*

Interior-point method

- *MSK_DPAR_CHECK_CONVEXITY_REL_TOL*
- *MSK_DPAR_INTPNT_CO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_CO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_REL_GAP*

- *MSK_DPAR_INTPNT_NL_MERIT_BAL*
- *MSK_DPAR_INTPNT_NL_TOL_DFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_MU_RED*
- *MSK_DPAR_INTPNT_NL_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_NL_TOL_PFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_NL_TOL_REL_STEP*
- *MSK_DPAR_INTPNT_QO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_QO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_TOL_DFEAS*
- *MSK_DPAR_INTPNT_TOL_DSAFE*
- *MSK_DPAR_INTPNT_TOL_INFEAS*
- *MSK_DPAR_INTPNT_TOL_MU_RED*
- *MSK_DPAR_INTPNT_TOL_PATH*
- *MSK_DPAR_INTPNT_TOL_PFEAS*
- *MSK_DPAR_INTPNT_TOL_PSAFE*
- *MSK_DPAR_INTPNT_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_TOL_REL_STEP*
- *MSK_DPAR_INTPNT_TOL_STEP_SIZE*
- *MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL*
- *MSK_IPAR_BI_IGNORE_MAX_ITER*
- *MSK_IPAR_BI_IGNORE_NUM_ERROR*
- *MSK_IPAR_INTPNT_BASIS*
- *MSK_IPAR_INTPNT_DIFF_STEP*
- *MSK_IPAR_INTPNT_HOTSTART*
- *MSK_IPAR_INTPNT_MAX_ITERATIONS*
- *MSK_IPAR_INTPNT_MAX_NUM_COR*
- *MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS*
- *MSK_IPAR_INTPNT_OFF_COL_TRH*
- *MSK_IPAR_INTPNT_ORDER_METHOD*
- *MSK_IPAR_INTPNT_REGULARIZATION_USE*
- *MSK_IPAR_INTPNT_SCALING*
- *MSK_IPAR_INTPNT_SOLVE_FORM*
- *MSK_IPAR_INTPNT_STARTING_POINT*
- *MSK_IPAR_LOG_INTPNT*

License manager

- *MSK_IPAR_CACHE_LICENSE*
- *MSK_IPAR_LICENSE_DEBUG*
- *MSK_IPAR_LICENSE_PAUSE_TIME*
- *MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS*
- *MSK_IPAR_LICENSE_TRH_EXPIRY_WRN*
- *MSK_IPAR_LICENSE_WAIT*

Logging

- *MSK_IPAR_LOG*
- *MSK_IPAR_LOG_ANA_PRO*
- *MSK_IPAR_LOG_BI*
- *MSK_IPAR_LOG_BI_FREQ*
- *MSK_IPAR_LOG_CUT_SECOND_OPT*
- *MSK_IPAR_LOG_EXPAND*
- *MSK_IPAR_LOG_FEAS_REPAIR*
- *MSK_IPAR_LOG_FILE*
- *MSK_IPAR_LOG_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_LOG_ORDER*
- *MSK_IPAR_LOG_PRESOLVE*
- *MSK_IPAR_LOG_RESPONSE*
- *MSK_IPAR_LOG_SENSITIVITY*
- *MSK_IPAR_LOG_SENSITIVITY_OPT*
- *MSK_IPAR_LOG_SIM*
- *MSK_IPAR_LOG_SIM_FREQ*
- *MSK_IPAR_LOG_STORAGE*

Mixed-integer optimization

- *MSK_DPAR_MIO_DISABLE_TERM_TIME*
- *MSK_DPAR_MIO_MAX_TIME*
- *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*
- *MSK_DPAR_MIO_REL_GAP_CONST*
- *MSK_DPAR_MIO_TOL_ABS_GAP*
- *MSK_DPAR_MIO_TOL_ABS_RELAX_INT*

- *MSK_DPAR_MIO_TOL_FEAS*
- *MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT*
- *MSK_DPAR_MIO_TOL_REL_GAP*
- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_MIO_BRANCH_DIR*
- *MSK_IPAR_MIO_CONSTRUCT_SOL*
- *MSK_IPAR_MIO_CUT_CLIQUE*
- *MSK_IPAR_MIO_CUT_CMIR*
- *MSK_IPAR_MIO_CUT_GMI*
- *MSK_IPAR_MIO_CUT_IMPLIED_BOUND*
- *MSK_IPAR_MIO_CUT_KNAPSACK_COVER*
- *MSK_IPAR_MIO_CUT_SELECTION_LEVEL*
- *MSK_IPAR_MIO_HEURISTIC_LEVEL*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_RELAXS*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *MSK_IPAR_MIO_NODE_OPTIMIZER*
- *MSK_IPAR_MIO_NODE_SELECTION*
- *MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE*
- *MSK_IPAR_MIO_PROBING_LEVEL*
- *MSK_IPAR_MIO_RINS_MAX_NODES*
- *MSK_IPAR_MIO_ROOT_OPTIMIZER*
- *MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL*
- *MSK_IPAR_MIO_VB_DETECTION_LEVEL*

Nonlinear convex method

- *MSK_DPAR_INTPNT_NL_MERIT_BAL*
- *MSK_DPAR_INTPNT_NL_TOL_DFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_MU_RED*
- *MSK_DPAR_INTPNT_NL_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_NL_TOL_PFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_NL_TOL_REL_STEP*
- *MSK_DPAR_INTPNT_TOL_INFEAS*
- *MSK_IPAR_CHECK_CONVEXITY*
- *MSK_IPAR_LOG_CHECK_CONVEXITY*

Output information

- *MSK_IPAR_INFEAS_REPORT_LEVEL*
- *MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS*
- *MSK_IPAR_LICENSE_TRH_EXPIRY_WRN*
- *MSK_IPAR_LOG*
- *MSK_IPAR_LOG_BI*
- *MSK_IPAR_LOG_BI_FREQ*
- *MSK_IPAR_LOG_CUT_SECOND_OPT*
- *MSK_IPAR_LOG_EXPAND*
- *MSK_IPAR_LOG_FEAS_REPAIR*
- *MSK_IPAR_LOG_FILE*
- *MSK_IPAR_LOG_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_MIO*
- *MSK_IPAR_LOG_MIO_FREQ*
- *MSK_IPAR_LOG_ORDER*
- *MSK_IPAR_LOG_RESPONSE*
- *MSK_IPAR_LOG_SENSITIVITY*
- *MSK_IPAR_LOG_SENSITIVITY_OPT*
- *MSK_IPAR_LOG_SIM*
- *MSK_IPAR_LOG_SIM_FREQ*
- *MSK_IPAR_LOG_SIM_MINOR*
- *MSK_IPAR_LOG_STORAGE*
- *MSK_IPAR_MAX_NUM_WARNINGS*

Overall solver

- *MSK_IPAR_BI_CLEAN_OPTIMIZER*
- *MSK_IPAR_INFEAS_PREFER_PRIMAL*
- *MSK_IPAR_LICENSE_WAIT*
- *MSK_IPAR_MIO_MODE*
- *MSK_IPAR_OPTIMIZER*
- *MSK_IPAR_PRESOLVE_LEVEL*
- *MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS*
- *MSK_IPAR_PRESOLVE_USE*
- *MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER*
- *MSK_IPAR_SENSITIVITY_ALL*
- *MSK_IPAR_SENSITIVITY_OPTIMIZER*
- *MSK_IPAR_SENSITIVITY_TYPE*

- *MSK_IPAR_SOLUTION_CALLBACK*

Overall system

- *MSK_IPAR_AUTO_UPDATE_SOL_INFO*
- *MSK_IPAR_INTPNT_MULTI_THREAD*
- *MSK_IPAR_LICENSE_WAIT*
- *MSK_IPAR_LOG_STORAGE*
- *MSK_IPAR_MIO_MT_USER_CB*
- *MSK_IPAR_MT_SPINCOUNT*
- *MSK_IPAR_NUM_THREADS*
- *MSK_IPAR_REMOVE_UNUSED_SOLUTIONS*
- *MSK_IPAR_TIMING_LEVEL*
- *MSK_SPAR_REMOTE_ACCESS_TOKEN*

Presolve

- *MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP*
- *MSK_DPAR_PRESOLVE_TOL_AIJ*
- *MSK_DPAR_PRESOLVE_TOL_REL_LINDEP*
- *MSK_DPAR_PRESOLVE_TOL_S*
- *MSK_DPAR_PRESOLVE_TOL_X*
- *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL*
- *MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES*
- *MSK_IPAR_PRESOLVE_LEVEL*
- *MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH*
- *MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH*
- *MSK_IPAR_PRESOLVE_LINDEP_USE*
- *MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS*
- *MSK_IPAR_PRESOLVE_USE*

Primal simplex

- *MSK_IPAR_SIM_PRIMAL_CRASH*
- *MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION*
- *MSK_IPAR_SIM_PRIMAL_SELECTION*

Progress callback

- *MSK_IPAR_SOLUTION_CALLBACK*

Simplex optimizer

- *MSK_DPAR_BASIS_REL_TOL_S*
- *MSK_DPAR_BASIS_TOL_S*
- *MSK_DPAR_BASIS_TOL_X*
- *MSK_DPAR_SIM_LU_TOL_REL_PIV*
- *MSK_DPAR_SIMPLEX_ABS_TOL_PIV*
- *MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE*
- *MSK_IPAR_LOG_SIM*
- *MSK_IPAR_LOG_SIM_FREQ*
- *MSK_IPAR_LOG_SIM_MINOR*
- *MSK_IPAR_SENSITIVITY_OPTIMIZER*
- *MSK_IPAR_SIM_BASIS_FACTOR_USE*
- *MSK_IPAR_SIM_DEGEN*
- *MSK_IPAR_SIM_DUAL_PHASEONE_METHOD*
- *MSK_IPAR_SIM_EXPLOIT_DUPVEC*
- *MSK_IPAR_SIM_HOTSTART*
- *MSK_IPAR_SIM_HOTSTART_LU*
- *MSK_IPAR_SIM_MAX_ITERATIONS*
- *MSK_IPAR_SIM_MAX_NUM_SETBACKS*
- *MSK_IPAR_SIM_NON_SINGULAR*
- *MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD*
- *MSK_IPAR_SIM_REFACTOR_FREQ*
- *MSK_IPAR_SIM_REFORMULATION*
- *MSK_IPAR_SIM_SAVE_LU*
- *MSK_IPAR_SIM_SCALING*
- *MSK_IPAR_SIM_SCALING_METHOD*
- *MSK_IPAR_SIM_SOLVE_FORM*
- *MSK_IPAR_SIM_STABILITY_PRIORITY*
- *MSK_IPAR_SIM_SWITCH_OPTIMIZER*

Solution input/output

- *MSK_IPAR_INFEAS_REPORT_AUTO*
- *MSK_IPAR_SOL_FILTER_KEEP_BASIC*
- *MSK_IPAR_SOL_FILTER_KEEP_RANGED*
- *MSK_IPAR_SOL_READ_NAME_WIDTH*
- *MSK_IPAR_SOL_READ_WIDTH*
- *MSK_IPAR_WRITE_BAS_CONSTRAINTS*
- *MSK_IPAR_WRITE_BAS_HEAD*

- *MSK_IPAR_WRITE_BAS_VARIABLES*
- *MSK_IPAR_WRITE_INT_CONSTRAINTS*
- *MSK_IPAR_WRITE_INT_HEAD*
- *MSK_IPAR_WRITE_INT_VARIABLES*
- *MSK_IPAR_WRITE_SOL_BARVARIABLES*
- *MSK_IPAR_WRITE_SOL_CONSTRAINTS*
- *MSK_IPAR_WRITE_SOL_HEAD*
- *MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES*
- *MSK_IPAR_WRITE_SOL_VARIABLES*
- *MSK_SPAR_BAS_SOL_FILE_NAME*
- *MSK_SPAR_INT_SOL_FILE_NAME*
- *MSK_SPAR_ITR_SOL_FILE_NAME*
- *MSK_SPAR_SOL_FILTER_XC_LOW*
- *MSK_SPAR_SOL_FILTER_XC_UPR*
- *MSK_SPAR_SOL_FILTER_XX_LOW*
- *MSK_SPAR_SOL_FILTER_XX_UPR*

Termination criteria

- *MSK_DPAR_BASIS_REL_TOL_S*
- *MSK_DPAR_BASIS_TOL_S*
- *MSK_DPAR_BASIS_TOL_X*
- *MSK_DPAR_INTPNT_CO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_CO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_CO_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_NL_TOL_DFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_MU_RED*
- *MSK_DPAR_INTPNT_NL_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_NL_TOL_PFEAS*
- *MSK_DPAR_INTPNT_NL_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_QO_TOL_DFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_INFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_MU_RED*
- *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*
- *MSK_DPAR_INTPNT_QO_TOL_PFEAS*
- *MSK_DPAR_INTPNT_QO_TOL_REL_GAP*
- *MSK_DPAR_INTPNT_TOL_DFEAS*

- *MSK_DPAR_INTPNT_TOL_INFEAS*
- *MSK_DPAR_INTPNT_TOL_MU_RED*
- *MSK_DPAR_INTPNT_TOL_PFEAS*
- *MSK_DPAR_INTPNT_TOL_REL_GAP*
- *MSK_DPAR_LOWER_OBJ_CUT*
- *MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH*
- *MSK_DPAR_MIO_DISABLE_TERM_TIME*
- *MSK_DPAR_MIO_MAX_TIME*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*
- *MSK_DPAR_MIO_REL_GAP_CONST*
- *MSK_DPAR_MIO_TOL_REL_GAP*
- *MSK_DPAR_OPTIMIZER_MAX_TIME*
- *MSK_DPAR_UPPER_OBJ_CUT*
- *MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH*
- *MSK_IPAR_BI_MAX_ITERATIONS*
- *MSK_IPAR_INTPNT_MAX_ITERATIONS*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *MSK_IPAR_SIM_MAX_ITERATIONS*

Other

- *MSK_IPAR_COMPRESS_STATFILE*

11.3 Parameters (alphabetical list sorted by type)

- *Double parameters*
- *Integer parameters*
- *String parameters*

11.3.1 Double parameters

`MSK_DPAR_ANA_SOL_INFEAS_TOL`

If a constraint violates its bound with an amount larger than this value, the constraint name, index and violation will be printed by the solution analyzer.

Default 1e-6

Accepted [0.0; +inf]

Groups *Analysis*

`MSK_DPAR_BASIS_REL_TOL_S`

Maximum relative dual bound violation allowed in an optimal basic solution.

Default 1.0e-12

Accepted [0.0; +inf]

Groups *Simplex optimizer*, *Termination criteria*

MSK_DPAR_BASIS_TOL_S

Maximum absolute dual bound violation in an optimal basic solution.

Default 1.0e-6

Accepted [1.0e-9; +inf]

Groups *Simplex optimizer*, *Termination criteria*

MSK_DPAR_BASIS_TOL_X

Maximum absolute primal bound violation allowed in an optimal basic solution.

Default 1.0e-6

Accepted [1.0e-9; +inf]

Groups *Simplex optimizer*, *Termination criteria*

MSK_DPAR_CHECK_CONVEXITY_REL_TOL

This parameter controls when the full convexity check declares a problem to be non-convex. Increasing this tolerance relaxes the criteria for declaring the problem non-convex.

A problem is declared non-convex if negative (positive) pivot elements are detected in the Cholesky factor of a matrix which is required to be PSD (NSD). This parameter controls how much this non-negativity requirement may be violated.

If d_i is the pivot element for column i, then the matrix Q is considered to not be PSD if:

$$d_i \leq -|Q_{ii}|\texttt{check\_convexity\_rel\_tol}$$

Default 1e-10

Accepted [0; +inf]

Groups *Interior-point method*

MSK_DPAR_DATA_SYM_MAT_TOL

Absolute zero tolerance for elements in in suymmetric matrixes. If any value in a symmetric matrix is smaller than this parameter in absolute terms **MOSEK** will treat the values as zero and generate a warning.

Default 1.0e-12

Accepted [1.0e-16; 1.0e-6]

Groups *Data check*

MSK_DPAR_DATA_SYM_MAT_TOL_HUGE

An element in a symmetric matrix which is larger than this value in absolute size causes an error.

Default 1.0e20

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_DATA_SYM_MAT_TOL_LARGE

An element in a symmetric matrix which is larger than this value in absolute size causes a warning message to be printed.

Default 1.0e10

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_DATA_TOL_AIJ

Absolute zero tolerance for elements in A. If any value A_{ij} is smaller than this parameter in absolute terms **MOSEK** will treat the values as zero and generate a warning.

Default 1.0e-12

Accepted [1.0e-16; 1.0e-6]

Groups *Data check*

MSK_DPAR_DATA_TOL_AIJ_HUGE
An element in A which is larger than this value in absolute size causes an error.

Default 1.0e20

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_DATA_TOL_AIJ_LARGE
An element in A which is larger than this value in absolute size causes a warning message to be printed.

Default 1.0e10

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_DATA_TOL_BOUND_INF
Any bound which in absolute value is greater than this parameter is considered infinite.

Default 1.0e16

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_DATA_TOL_BOUND_WRN
If a bound value is larger than this value in absolute size, then a warning message is issued.

Default 1.0e8

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_DATA_TOL_C_HUGE
An element in c which is larger than the value of this parameter in absolute terms is considered to be huge and generates an error.

Default 1.0e16

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_DATA_TOL_CJ_LARGE
An element in c which is larger than this value in absolute terms causes a warning message to be printed.

Default 1.0e8

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_DATA_TOL_QIJ
Absolute zero tolerance for elements in Q matrices.

Default 1.0e-16

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_DATA_TOL_X
: Zero tolerance for constraints and variables i.e. if the distance between the lower and upper bound is less than this value, then the lower and upper bound is considered identical.

Default 1.0e-8

Accepted [0.0; +inf]

Groups *Data check*

MSK_DPAR_INTPNT_CO_TOL_DFEAS
: Dual feasibility tolerance used by the conic interior-point optimizer.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*, *Conic interior-point method*

See also *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*

MSK_DPAR_INTPNT_CO_TOL_INFEAS
: Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

Default 1.0e-10

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*, *Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_MU_RED
: Relative complementarity gap feasibility tolerance used by the conic interior-point optimizer.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*, *Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_NEAR_REL
: If **MOSEK** cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

Default 1000

Accepted [1.0; +inf]

Groups *Interior-point method*, *Termination criteria*, *Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_PFEAS
: Primal feasibility tolerance used by the conic interior-point optimizer.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*, *Conic interior-point method*

See also *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*

MSK_DPAR_INTPNT_CO_TOL_REL_GAP
: Relative gap termination tolerance used by the conic interior-point optimizer.

Default 1.0e-7

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*, *Conic interior-point method*

See also *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*

MSK_DPAR_INTPNT_NL_MERIT_BAL
Controls if the complementarity and infeasibility is converging to zero at about equal rates.

Default 1.0e-4

Accepted [0.0; 0.99]

Groups *Interior-point method*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_DFEAS
Dual feasibility tolerance used when a nonlinear model is solved.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_MU_RED
Relative complementarity gap tolerance for the nonlinear solver.

Default 1.0e-12

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_NEAR_REL
If the **MOSEK** nonlinear interior-point optimizer cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

Default 1000.0

Accepted [1.0; +inf]

Groups *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_PFEAS
Primal feasibility tolerance used when a nonlinear model is solved.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_REL_GAP
Relative gap termination tolerance for nonlinear problems.

Default 1.0e-6

Accepted [1.0e-14; +inf]

Groups *Termination criteria*, *Interior-point method*, *Nonlinear convex method*

MSK_DPAR_INTPNT_NL_TOL_REL_STEP
Relative step size to the boundary for general nonlinear optimization problems.

Default 0.995

Accepted [1.0e-4; 0.9999999]

Groups *Interior-point method*, *Nonlinear convex method*

MSK_DPAR_INTPNT_QO_TOL_DFEAS
Dual feasibility tolerance used when the interior-point optimizer is applied to a quadratic optimization problem..

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*

See also *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*

`MSK_DPAR_INTPNT_QO_TOL_INFEAS`

Controls when the conic interior-point optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

Default 1.0e-10

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*

`MSK_DPAR_INTPNT_QO_TOL_MU_RED`

Relative complementarity gap feasibility tolerance used when interior-point optimizer is applied to a quadratic optimization problem.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*

`MSK_DPAR_INTPNT_QO_TOL_NEAR_REL`

If **MOSEK** cannot compute a solution that has the prescribed accuracy, then it will multiply the termination tolerances with value of this parameter. If the solution then satisfies the termination criteria, then the solution is denoted near optimal, near feasible and so forth.

Default 1000

Accepted [1.0; +inf]

Groups *Interior-point method*, *Termination criteria*

`MSK_DPAR_INTPNT_QO_TOL_PFEAS`

Primal feasibility tolerance used when the interior-point optimizer is applied to a quadratic optimization problem.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*

See also *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*

`MSK_DPAR_INTPNT_QO_TOL_REL_GAP`

Relative gap termination tolerance used when the interior-point optimizer is applied to a quadratic optimization problem.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*

See also *MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*

`MSK_DPAR_INTPNT_TOL_DFEAS`

Dual feasibility tolerance used for linear optimization problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*

`MSK_DPAR_INTPNT_TOL_DSAFE`

Controls the initial dual starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it might be worthwhile to increase this value.

Default 1.0

Accepted [1.0e-4; +inf]

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_INFEAS

Controls when the optimizer declares the model primal or dual infeasible. A small number means the optimizer gets more conservative about declaring the model infeasible.

Default 1.0e-10

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*, *Nonlinear convex method*

MSK_DPAR_INTPNT_TOL_MU_RED

Relative complementarity gap tolerance for linear problems.

Default 1.0e-16

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*

MSK_DPAR_INTPNT_TOL_PATH

Controls how close the interior-point optimizer follows the central path. A large value of this parameter means the central is followed very closely. On numerical unstable problems it may be worthwhile to increase this parameter.

Default 1.0e-8

Accepted [0.0; 0.9999]

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_PFEAS

Primal feasibility tolerance used for linear optimization problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Groups *Interior-point method*, *Termination criteria*

MSK_DPAR_INTPNT_TOL_PSAFE

Controls the initial primal starting point used by the interior-point optimizer. If the interior-point optimizer converges slowly and/or the constraint or variable bounds are very large, then it may be worthwhile to increase this value.

Default 1.0

Accepted [1.0e-4; +inf]

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_REL_GAP

Relative gap termination tolerance for linear problems.

Default 1.0e-8

Accepted [1.0e-14; +inf]

Groups *Termination criteria*, *Interior-point method*

MSK_DPAR_INTPNT_TOL_REL_STEP

Relative step size to the boundary for linear and quadratic optimization problems.

Default 0.9999

Accepted [1.0e-4; 0.999999]

Groups *Interior-point method*

`MSK_DPAR_INTPNT_TOL_STEP_SIZE`

Minimal step size tolerance. If the step size falls below the value of this parameter, then the interior-point optimizer assumes that it is stalled. In other words the interior-point optimizer does not make any progress and therefore it is better stop.

Default 1.0e-6

Accepted [0.0; 1.0]

Groups *Interior-point method*

`MSK_DPAR_LOWER_OBJ_CUT`

If either a primal or dual feasible solution is found proving that the optimal objective value is outside, the interval [*MSK_DPAR_LOWER_OBJ_CUT*, *MSK_DPAR_UPPER_OBJ_CUT*], then **MOSEK** is terminated.

Default -1.0e30

Accepted [-inf; +inf]

Groups *Termination criteria*

See also *MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH*

`MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH`

If the lower objective cut is less than the value of this parameter value, then the lower objective cut i.e. *MSK_DPAR_LOWER_OBJ_CUT* is treated as $-\infty$.

Default -0.5e30

Accepted [-inf; +inf]

Groups *Termination criteria*

`MSK_DPAR_MIO_DISABLE_TERM_TIME`

This parameter specifies the number of seconds n during which the termination criteria governed by

- *MSK_IPAR_MIO_MAX_NUM_RELAXS*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*
- *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*

is disabled since the beginning of the optimization.

A negative value is identical to infinity i.e. the termination criteria are never checked.

Default -1.0

Accepted [-inf; +inf]

Groups *Mixed-integer optimization*, *Termination criteria*

See also *MSK_IPAR_MIO_MAX_NUM_RELAXS*, *MSK_IPAR_MIO_MAX_NUM_BRANCHES*, *MSK_DPAR_MIO_NEAR_TOL_ABS_GAP*, *MSK_DPAR_MIO_NEAR_TOL_REL_GAP*

`MSK_DPAR_MIO_MAX_TIME`

This parameter limits the maximum time spent by the mixed-integer optimizer. A negative number means infinity.

Default -1.0

Accepted [-inf; +inf]

Groups *Mixed-integer optimization*, *Termination criteria*

`MSK_DPAR_MIO_NEAR_TOL_ABS_GAP`
Relaxed absolute optimality tolerance employed by the mixed-integer optimizer. This termination criteria is delayed. See *MSK_DPAR_MIO_DISABLE_TERM_TIME* for details.

Default 0.0

Accepted [0.0; +inf]

Groups *Mixed-integer optimization*

See also *MSK_DPAR_MIO_DISABLE_TERM_TIME*

`MSK_DPAR_MIO_NEAR_TOL_REL_GAP`
The mixed-integer optimizer is terminated when this tolerance is satisfied. This termination criteria is delayed. See *MSK_DPAR_MIO_DISABLE_TERM_TIME* for details.

Default 1.0e-3

Accepted [0.0; +inf]

Groups *Mixed-integer optimization*, *Termination criteria*

See also *MSK_DPAR_MIO_DISABLE_TERM_TIME*

`MSK_DPAR_MIO_REL_GAP_CONST`
This value is used to compute the relative gap for the solution to an integer optimization problem.

Default 1.0e-10

Accepted [1.0e-15; +inf]

Groups *Mixed-integer optimization*, *Termination criteria*

`MSK_DPAR_MIO_TOL_ABS_GAP`
Absolute optimality tolerance employed by the mixed-integer optimizer.

Default 0.0

Accepted [0.0; +inf]

Groups *Mixed-integer optimization*

`MSK_DPAR_MIO_TOL_ABS_RELAX_INT`
Absolute integer feasibility tolerance. If the distance to the nearest integer is less than this tolerance then an integer constraint is assumed to be satisfied.

Default 1.0e-5

Accepted [1e-9; +inf]

Groups *Mixed-integer optimization*

`MSK_DPAR_MIO_TOL_FEAS`
Feasibility tolerance for mixed integer solver.

Default 1.0e-6

Accepted [1e-9; 1e-3]

Groups *Mixed-integer optimization*

`MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT`
If the relative improvement of the dual bound is smaller than this value, the solver will terminate the root cut generation. A value of 0.0 means that the value is selected automatically.

Default 0.0

Accepted [0.0; 1.0]

Groups *Mixed-integer optimization*

`MSK_DPAR_MIO_TOL_REL_GAP`
Relative optimality tolerance employed by the mixed-integer optimizer.

Default 1.0e-4

Accepted [0.0; +inf]

Groups *Mixed-integer optimization*, *Termination criteria*

`MSK_DPAR_OPTIMIZER_MAX_TIME`
Maximum amount of time the optimizer is allowed to spent on the optimization. A negative number means infinity.

Default -1.0

Accepted [-inf; +inf]

Groups *Termination criteria*

`MSK_DPAR_PRESOLVE_TOL_ABS_LINDEP`
Absolute tolerance employed by the linear dependency checker.

Default 1.0e-6

Accepted [0.0; +inf]

Groups *Presolve*

`MSK_DPAR_PRESOLVE_TOL_AIJ`
Absolute zero tolerance employed for a_{ij} in the presolve.

Default 1.0e-12

Accepted [1.0e-15; +inf]

Groups *Presolve*

`MSK_DPAR_PRESOLVE_TOL_REL_LINDEP`
Relative tolerance employed by the linear dependency checker.

Default 1.0e-10

Accepted [0.0; +inf]

Groups *Presolve*

`MSK_DPAR_PRESOLVE_TOL_S`
Absolute zero tolerance employed for s_i in the presolve.

Default 1.0e-8

Accepted [0.0; +inf]

Groups *Presolve*

`MSK_DPAR_PRESOLVE_TOL_X`
Absolute zero tolerance employed for x_j in the presolve.

Default 1.0e-8

Accepted [0.0; +inf]

Groups *Presolve*

`MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL`
This parameter determines when columns are dropped in incomplete Cholesky factorization during reformulation of quadratic problems.

Default 1e-15

Accepted [0; +inf]

Groups *Interior-point method*

`MSK_DPAR_SEMIDEFINITE_TOL_APPROX`
Tolerance to define a matrix to be positive semidefinite.

Default 1.0e-10

Accepted [1.0e-15; +inf]

Groups *Data check*

MSK_DPAR_SIM_LU_TOL_REL_PIV

Relative pivot tolerance employed when computing the LU factorization of the basis in the simplex optimizers and in the basis identification procedure.

A value closer to 1.0 generally improves numerical stability but typically also implies an increase in the computational work.

Default 0.01

Accepted [1.0e-6; 0.999999]

Groups *Basis identification*, *Simplex optimizer*

MSK_DPAR_SIMPLEX_ABS_TOL_PIV

Absolute pivot tolerance employed by the simplex optimizers.

Default 1.0e-7

Accepted [1.0e-12; +inf]

Groups *Simplex optimizer*

MSK_DPAR_UPPER_OBJ_CUT

If either a primal or dual feasible solution is found proving that the optimal objective value is outside, the interval [*MSK_DPAR_LOWER_OBJ_CUT*, *MSK_DPAR_UPPER_OBJ_CUT*], then **MOSEK** is terminated.

Default 1.0e30

Accepted [-inf; +inf]

Groups *Termination criteria*

See also *MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH*

MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH

If the upper objective cut is greater than the value of this parameter, then the upper objective cut *MSK_DPAR_UPPER_OBJ_CUT* is treated as ∞.

Default 0.5e30

Accepted [-inf; +inf]

Groups *Termination criteria*

11.3.2 Integer parameters

MSK_IPAR_ANA_SOL_BASIS

Controls whether the basis matrix is analyzed in solution analyzer.

Default *ON*

Accepted *ON*, *OFF*

Groups *Analysis*

MSK_IPAR_ANA_SOL_PRINT_VIOLATED

Controls whether a list of violated constraints is printed.

All constraints violated by more than the value set by the parameter *MSK_DPAR_ANA_SOL_INFEAS_TOL* will be printed.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Analysis*

MSK_IPAR_AUTO_SORT_A_BEFORE_OPT
Controls whether the elements in each column of A are sorted before an optimization is performed. This is not required but makes the optimization more deterministic.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Debugging*

MSK_IPAR_AUTO_UPDATE_SOL_INFO
Controls whether the solution information items are automatically updated after an optimization is performed.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Overall system*

MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE
If a slack variable is in the basis, then the corresponding column in the basis is a unit vector with -1 in the right position. However, if this parameter is set to *MSK_ON*, -1 is replaced by 1.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Simplex optimizer*

MSK_IPAR_BI_CLEAN_OPTIMIZER
Controls which simplex optimizer is used in the clean-up phase.

Default *FREE*

Accepted *FREE*, *INTPNT*, *CONIC*, *PRIMAL_SIMPLEX*, *DUAL_SIMPLEX*, *FREE_SIMPLEX*, *MIXED_INT*

Groups *Basis identification*, *Overall solver*

MSK_IPAR_BI_IGNORE_MAX_ITER
If the parameter *MSK_IPAR_INTPNT_BASIS* has the value *MSK_BI_NO_ERROR* and the interior-point optimizer has terminated due to maximum number of iterations, then basis identification is performed if this parameter has the value *MSK_ON*.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Interior-point method*, *Basis identification*

MSK_IPAR_BI_IGNORE_NUM_ERROR
If the parameter *MSK_IPAR_INTPNT_BASIS* has the value *MSK_BI_NO_ERROR* and the interior-point optimizer has terminated due to a numerical problem, then basis identification is performed if this parameter has the value *MSK_ON*.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Interior-point method*, *Basis identification*

MSK_IPAR_BI_MAX_ITERATIONS
Controls the maximum number of simplex iterations allowed to optimize a basis after the basis identification.

Default 1000000

Accepted [0; +inf]

Groups *Basis identification*, *Termination criteria*

MSK_IPAR_CACHE_LICENSE

Specifies if the license is kept checked out for the lifetime of the mosek environment (*MSK_ON*) or returned to the server immediately after the optimization (*MSK_OFF*).

By default the license is checked out for the lifetime of the **MOSEK** environment by the first call to the optimizer.

Check-in and check-out of licenses have an overhead. Frequent communication with the license server should be avoided.

Default *ON*

Accepted *ON*, *OFF*

Groups *License manager*

MSK_IPAR_CHECK_CONVEXITY

Specify the level of convexity check on quadratic problems.

Default *FULL*

Accepted *NONE*, *SIMPLE*, *FULL*

Groups *Data check*, *Nonlinear convex method*

MSK_IPAR_COMPRESS_STATFILE

Control compression of stat files.

Default *ON*

Accepted *ON*, *OFF*

MSK_IPAR_INFEAS_GENERIC_NAMES

Controls whether generic names are used when an infeasible subproblem is created.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Infeasibility report*

MSK_IPAR_INFEAS_PREFER_PRIMAL

If both certificates of primal and dual infeasibility are supplied then only the primal is used when this option is turned on.

Default *ON*

Accepted *ON*, *OFF*

Groups *Overall solver*

MSK_IPAR_INFEAS_REPORT_AUTO

Controls whether an infeasibility report is automatically produced after the optimization if the problem is primal or dual infeasible.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_INFEAS_REPORT_LEVEL

Controls the amount of information presented in an infeasibility report. Higher values imply more information.

Default 1

Accepted [0; +inf]

Groups *Infeasibility report*, *Output information*

MSK_IPAR_INTPNT_BASIS
Controls whether the interior-point optimizer also computes an optimal basis.

Default *ALWAYS*

Accepted *NEVER*, *ALWAYS*, *NO_ERROR*, *IF_FEASIBLE*, *RESERVERED*

Groups *Interior-point method*, *Basis identification*

See also *MSK_IPAR_BI_IGNORE_MAX_ITER*, *MSK_IPAR_BI_IGNORE_NUM_ERROR*, *MSK_IPAR_BI_MAX_ITERATIONS*, *MSK_IPAR_BI_CLEAN_OPTIMIZER*

MSK_IPAR_INTPNT_DIFF_STEP
Controls whether different step sizes are allowed in the primal and dual space.

Default *ON*

Accepted

- *ON*: Different step sizes are allowed.
- *OFF*: Different step sizes are not allowed.

Groups *Interior-point method*

MSK_IPAR_INTPNT_HOTSTART
Currently not in use.

Default *NONE*

Accepted *NONE*, *PRIMAL*, *DUAL*, *PRIMAL_DUAL*

Groups *Interior-point method*

MSK_IPAR_INTPNT_MAX_ITERATIONS
Controls the maximum number of iterations allowed in the interior-point optimizer.

Default 400

Accepted [0; +inf]

Groups *Interior-point method*, *Termination criteria*

MSK_IPAR_INTPNT_MAX_NUM_COR
Controls the maximum number of correctors allowed by the multiple corrector procedure. A negative value means that **MOSEK** is making the choice.

Default -1

Accepted [-1; +inf]

Groups *Interior-point method*

MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS
Maximum number of steps to be used by the iterative refinement of the search direction. A negative value implies that the optimizer chooses the maximum number of iterative refinement steps.

Default -1

Accepted [-inf; +inf]

Groups *Interior-point method*

MSK_IPAR_INTPNT_MULTI_THREAD
Controls whether the interior-point optimizers are allowed to employ multiple threads if more threads is available.

Default *ON*

Accepted *ON*, *OFF*

Groups *Overall system*

MSK_IPAR_INTPNT_OFF_COL_TRH

Controls how many offending columns are detected in the Jacobian of the constraint matrix.

0	no detection
1	aggressive detection
> 1	higher values mean less aggressive detection

Default 40

Accepted [0; +inf]

Groups *Interior-point method*

MSK_IPAR_INTPNT_ORDER_METHOD

Controls the ordering strategy used by the interior-point optimizer when factorizing the Newton equation system.

Default *FREE*

Accepted *FREE*, *APPMINLOC*, *EXPERIMENTAL*, *TRY_GRAPHPAR*, *FORCE_GRAPHPAR*, *NONE*

Groups *Interior-point method*

MSK_IPAR_INTPNT_REGULARIZATION_USE

Controls whether regularization is allowed.

Default *ON*

Accepted *ON*, *OFF*

Groups *Interior-point method*

MSK_IPAR_INTPNT_SCALING

Controls how the problem is scaled before the interior-point optimizer is used.

Default *FREE*

Accepted *FREE*, *NONE*, *MODERATE*, *AGGRESSIVE*

Groups *Interior-point method*

MSK_IPAR_INTPNT_SOLVE_FORM

Controls whether the primal or the dual problem is solved.

Default *FREE*

Accepted *FREE*, *PRIMAL*, *DUAL*

Groups *Interior-point method*

MSK_IPAR_INTPNT_STARTING_POINT

Starting point used by the interior-point optimizer.

Default *FREE*

Accepted *FREE*, *GUESS*, *CONSTANT*, *SATISFY_BOUNDS*

Groups *Interior-point method*

MSK_IPAR_LICENSE_DEBUG

This option is used to turn on debugging of the license manager.

Default *OFF*

Accepted *ON*, *OFF*

Groups *License manager*

`MSK_IPAR_LICENSE_PAUSE_TIME`
If *MSK_IPAR_LICENSE_WAIT* = *MSK_ON* and no license is available, then **MOSEK** sleeps a number of milliseconds between each check of whether a license has become free.

Default 100

Accepted [0; 1000000]

Groups *License manager*

`MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS`
Controls whether license features expire warnings are suppressed.

Default *OFF*

Accepted *ON*, *OFF*

Groups *License manager*, *Output information*

`MSK_IPAR_LICENSE_TRH_EXPIRY_WRN`
If a license feature expires in a numbers days less than the value of this parameter then a warning will be issued.

Default 7

Accepted [0; +inf]

Groups *License manager*, *Output information*

`MSK_IPAR_LICENSE_WAIT`
If all licenses are in use **MOSEK** returns with an error code. However, by turning on this parameter **MOSEK** will wait for an available license.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Overall solver*, *Overall system*, *License manager*

`MSK_IPAR_LOG`
Controls the amount of log information. The value 0 implies that all log information is suppressed. A higher level implies that more information is logged.

Please note that if a task is employed to solve a sequence of optimization problems the value of this parameter is reduced by the value of *MSK_IPAR_LOG_CUT_SECOND_OPT* for the second and any subsequent optimizations.

Default 10

Accepted [0; +inf]

Groups *Output information*, *Logging*

See also *MSK_IPAR_LOG_CUT_SECOND_OPT*

`MSK_IPAR_LOG_ANA_PRO`
Controls amount of output from the problem analyzer.

Default 1

Accepted [0; +inf]

Groups *Analysis*, *Logging*

`MSK_IPAR_LOG_BI`
Controls the amount of output printed by the basis identification procedure. A higher level implies that more information is logged.

Default 1

Accepted [0; +inf]

Groups *Basis identification*, *Output information*, *Logging*

MSK_IPAR_LOG_BI_FREQ
Controls how frequent the optimizer outputs information about the basis identification and how frequent the user-defined callback function is called.

Default 2500

Accepted [0; +inf]

Groups *Basis identification*, *Output information*, *Logging*

MSK_IPAR_LOG_CHECK_CONVEXITY
Controls logging in convexity check on quadratic problems. Set to a positive value to turn logging on. If a quadratic coefficient matrix is found to violate the requirement of PSD (NSD) then a list of negative (positive) pivot elements is printed. The absolute value of the pivot elements is also shown.

Default 0

Accepted [0; +inf]

Groups *Data check*, *Nonlinear convex method*

MSK_IPAR_LOG_CUT_SECOND_OPT
If a task is employed to solve a sequence of optimization problems, then the value of the log levels is reduced by the value of this parameter. E.g *MSK_IPAR_LOG* and *MSK_IPAR_LOG_SIM* are reduced by the value of this parameter for the second and any subsequent optimizations.

Default 1

Accepted [0; +inf]

Groups *Output information*, *Logging*

See also *MSK_IPAR_LOG*, *MSK_IPAR_LOG_INTPNT*, *MSK_IPAR_LOG_MIO*, *MSK_IPAR_LOG_SIM*

MSK_IPAR_LOG_EXPAND
Controls the amount of logging when a data item such as the maximum number constrains is expanded.

Default 0

Accepted [0; +inf]

Groups *Output information*, *Logging*

MSK_IPAR_LOG_FEAS_REPAIR
Controls the amount of output printed when performing feasibility repair. A value higher than one means extensive logging.

Default 1

Accepted [0; +inf]

Groups *Output information*, *Logging*

MSK_IPAR_LOG_FILE
If turned on, then some log info is printed when a file is written or read.

Default 1

Accepted [0; +inf]

Groups *Data input/output*, *Output information*, *Logging*

MSK_IPAR_LOG_INFEAS_ANA
Controls amount of output printed by the infeasibility analyzer procedures. A higher level implies that more information is logged.

Default 1

Accepted [0; +inf]

Groups *Infeasibility report*, *Output information*, *Logging*

MSK_IPAR_LOG_INTPNT
Controls amount of output printed by the interior-point optimizer. A higher level implies that more information is logged.

Default 1

Accepted [0; +inf]

Groups *Interior-point method*, *Output information*, *Logging*

MSK_IPAR_LOG_MIO
Controls the log level for the mixed-integer optimizer. A higher level implies that more information is logged.

Default 4

Accepted [0; +inf]

Groups *Mixed-integer optimization*, *Output information*, *Logging*

MSK_IPAR_LOG_MIO_FREQ
Controls how frequent the mixed-integer optimizer prints the log line. It will print line every time *MSK_IPAR_LOG_MIO_FREQ* relaxations have been solved.

Default 10

Accepted [-inf; +inf]

Groups *Mixed-integer optimization*, *Output information*, *Logging*

MSK_IPAR_LOG_ORDER
If turned on, then factor lines are added to the log.

Default 1

Accepted [0; +inf]

Groups *Output information*, *Logging*

MSK_IPAR_LOG_PRESOLVE
Controls amount of output printed by the presolve procedure. A higher level implies that more information is logged.

Default 1

Accepted [0; +inf]

Groups *Logging*

MSK_IPAR_LOG_RESPONSE
Controls amount of output printed when response codes are reported. A higher level implies that more information is logged.

Default 0

Accepted [0; +inf]

Groups *Output information*, *Logging*

MSK_IPAR_LOG_SENSITIVITY
Controls the amount of logging during the sensitivity analysis.

0. Means no logging information is produced.
1. Timing information is printed.
2. Sensitivity results are printed.

Default 1

Accepted [0; +inf]

Groups *Output information*, *Logging*

`MSK_IPAR_LOG_SENSITIVITY_OPT`

Controls the amount of logging from the optimizers employed during the sensitivity analysis. 0 means no logging information is produced.

Default 0

Accepted [0; +inf]

Groups *Output information*, *Logging*

`MSK_IPAR_LOG_SIM`

Controls amount of output printed by the simplex optimizer. A higher level implies that more information is logged.

Default 4

Accepted [0; +inf]

Groups *Simplex optimizer*, *Output information*, *Logging*

`MSK_IPAR_LOG_SIM_FREQ`

Controls how frequent the simplex optimizer outputs information about the optimization and how frequent the user-defined callback function is called.

Default 1000

Accepted [0; +inf]

Groups *Simplex optimizer*, *Output information*, *Logging*

`MSK_IPAR_LOG_SIM_MINOR`

Currently not in use.

Default 1

Accepted [0; +inf]

Groups *Simplex optimizer*, *Output information*

`MSK_IPAR_LOG_STORAGE`

When turned on, **MOSEK** prints messages regarding the storage usage and allocation.

Default 0

Accepted [0; +inf]

Groups *Output information*, *Overall system*, *Logging*

`MSK_IPAR_MAX_NUM_WARNINGS`

Each warning is shown a limit number times controlled by this parameter. A negative value is identical to infinite number of times.

Default 10

Accepted [-inf; +inf]

Groups *Output information*

`MSK_IPAR_MIO_BRANCH_DIR`

Controls whether the mixed-integer optimizer is branching up or down by default.

Default *FREE*

Accepted *FREE*, *UP*, *DOWN*, *NEAR*, *FAR*, *ROOT_LP*, *GUIDED*, *PSEUDOCOST*

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CONSTRUCT_SOL

If set to *MSK_ON* and all integer variables have been given a value for which a feasible mixed integer solution exists, then **MOSEK** generates an initial solution to the mixed integer problem by fixing all integer values and solving the remaining problem.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CLIQUE

Controls whether clique cuts should be generated.

Default *ON*

Accepted

- *ON*: Turns generation of this cut class on.
- *OFF*: Turns generation of this cut class off.

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CMIR

Controls whether mixed integer rounding cuts should be generated.

Default *ON*

Accepted

- *ON*: Turns generation of this cut class on.
- *OFF*: Turns generation of this cut class off.

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_GMI

Controls whether GMI cuts should be generated.

Default *ON*

Accepted

- *ON*: Turns generation of this cut class on.
- *OFF*: Turns generation of this cut class off.

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_IMPLIED_BOUND

Controls whether implied bound cuts should be generated.

Default *OFF*

Accepted

- *ON*: Turns generation of this cut class on.
- *OFF*: Turns generation of this cut class off.

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_KNAPSACK_COVER

Controls whether knapsack cover cuts should be generated.

Default *OFF*

Accepted

- *ON*: Turns generation of this cut class on.
- *OFF*: Turns generation of this cut class off.

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_SELECTION_LEVEL
Controls how aggressively generated cuts are selected to be included in the relaxation.

-1. The optimizer chooses the level of cut selection
0. Generated cuts less likely to be added to the relaxation
1. Cuts are more aggressively selected to be included in the relaxation

Default -1

Accepted [-1; +1]

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_HEURISTIC_LEVEL
Controls the heuristic employed by the mixed-integer optimizer to locate an initial good integer feasible solution. A value of zero means the heuristic is not used at all. A larger value than 0 means that a gradually more sophisticated heuristic is used which is computationally more expensive. A negative value implies that the optimizer chooses the heuristic. Normally a value around 3 to 5 should be optimal.

Default -1

Accepted [-inf; +inf]

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_MAX_NUM_BRANCHES
Maximum number of branches allowed during the branch and bound search. A negative value means infinite.

Default -1

Accepted [-inf; +inf]

Groups *Mixed-integer optimization*, *Termination criteria*

See also *MSK_DPAR_MIO_DISABLE_TERM_TIME*

MSK_IPAR_MIO_MAX_NUM_RELAXS
Maximum number of relaxations allowed during the branch and bound search. A negative value means infinite.

Default -1

Accepted [-inf; +inf]

Groups *Mixed-integer optimization*

See also *MSK_DPAR_MIO_DISABLE_TERM_TIME*

MSK_IPAR_MIO_MAX_NUM_SOLUTIONS
The mixed-integer optimizer can be terminated after a certain number of different feasible solutions has been located. If this parameter has the value $n > 0$, then the mixed-integer optimizer will be terminated when n feasible solutions have been located.

Default -1

Accepted [-inf; +inf]

Groups *Mixed-integer optimization*, *Termination criteria*

See also *MSK_DPAR_MIO_DISABLE_TERM_TIME*

MSK_IPAR_MIO_MODE
Controls whether the optimizer includes the integer restrictions when solving a (mixed) integer optimization problem.

Default *SATISFIED*

Accepted *IGNORED*, *SATISFIED*

Groups *Overall solver*

MSK_IPAR_MIO_MT_USER_CB

If true user callbacks are called from each thread used by mixed-integer optimizer. Otherwise it is only called from a single thread.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Overall system*

MSK_IPAR_MIO_NODE_OPTIMIZER

Controls which optimizer is employed at the non-root nodes in the mixed-integer optimizer.

Default *FREE*

Accepted *FREE*, *INTPNT*, *CONIC*, *PRIMAL_SIMPLEX*, *DUAL_SIMPLEX*, *FREE_SIMPLEX*, *MIXED_INT*

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_NODE_SELECTION

Controls the node selection strategy employed by the mixed-integer optimizer.

Default *FREE*

Accepted *FREE*, *FIRST*, *BEST*, *WORST*, *HYBRID*, *PSEUDO*

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE

Enables or disables perspective reformulation in presolve.

Default *ON*

Accepted *ON*, *OFF*

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_PROBING_LEVEL

Controls the amount of probing employed by the mixed-integer optimizer in presolve.

-1. The optimizer chooses the level of probing employed

0. Probing is disabled

1. A low amount of probing is employed

2. A medium amount of probing is employed

3. A high amount of probing is employed

Default -1

Accepted [-1; 3]

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_RINS_MAX_NODES

Controls the maximum number of nodes allowed in each call to the RINS heuristic. The default value of -1 means that the value is determined automatically. A value of zero turns off the heuristic.

Default -1

Accepted [-1; +inf]

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_ROOT_OPTIMIZER

Controls which optimizer is employed at the root node in the mixed-integer optimizer.

Default *FREE*

Accepted *FREE*, *INTPNT*, *CONIC*, *PRIMAL_SIMPLEX*, *DUAL_SIMPLEX*, *FREE_SIMPLEX*, *MIXED_INT*

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_ROOT_REPEAT_PRESOLVE_LEVEL

Controls whether presolve can be repeated at root node.

- -1 The optimizer chooses whether presolve is repeated
- 0 Never repeat presolve
- 1 Always repeat presolve

Default -1

Accepted [-1; 1]

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_VB_DETECTION_LEVEL

Controls how much effort is put into detecting variable bounds.

-1. The optimizer chooses
0. No variable bounds are detected
1. Only detect variable bounds that are directly represented in the problem
2. Detect variable bounds in probing

Default -1

Accepted [-1; +2]

Groups *Mixed-integer optimization*

MSK_IPAR_MT_SPINCOUNT

Set the number of iterations to spin before sleeping.

Default 0

Accepted [0; 1000000000]

Groups *Overall system*

MSK_IPAR_NUM_THREADS

Controls the number of threads employed by the optimizer. If set to 0 the number of threads used will be equal to the number of cores detected on the machine.

Default 0

Accepted [0; +inf]

Groups *Overall system*

MSK_IPAR_OPF_MAX_TERMS_PER_LINE

The maximum number of terms (linear and quadratic) per line when an OPF file is written.

Default 5

Accepted [0; +inf]

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_HEADER

Write a text header with date and **MOSEK** version in an OPF file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_HINTS
Write a hint section with problem dimensions in the beginning of an OPF file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_PARAMETERS
Write a parameter section in an OPF file.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_PROBLEM
Write objective, constraints, bounds etc. to an OPF file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_SOL_BAS
If *MSK_IPAR_OPF_WRITE_SOLUTIONS* is *MSK_ON* and a basic solution is defined, include the basic solution in OPF files.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_SOL_ITG
If *MSK_IPAR_OPF_WRITE_SOLUTIONS* is *MSK_ON* and an integer solution is defined, write the integer solution in OPF files.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_SOL_ITR
If *MSK_IPAR_OPF_WRITE_SOLUTIONS* is *MSK_ON* and an interior solution is defined, write the interior solution in OPF files.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_SOLUTIONS
Enable inclusion of solutions in the OPF files.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_OPTIMIZER
The parameter controls which optimizer is used to optimize the task.

Default *FREE*

Accepted *FREE*, *INTPNT*, *CONIC*, *PRIMAL_SIMPLEX*, *DUAL_SIMPLEX*, *FREE_SIMPLEX*, *MIXED_INT*

Groups *Overall solver*

MSK_IPAR_PARAM_READ_CASE_NAME

If turned on, then names in the parameter file are case sensitive.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_PARAM_READ_IGN_ERROR

If turned on, then errors in parameter settings is ignored.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL

Controls the maximum amount of fill-in that can be created by one pivot in the elimination phase of the presolve. A negative value means the parameter value is selected automatically.

Default -1

Accepted [-inf; +inf]

Groups *Presolve*

MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES

Control the maximum number of times the eliminator is tried. A negative value implies **MOSEK** decides.

Default -1

Accepted [-inf; +inf]

Groups *Presolve*

MSK_IPAR_PRESOLVE_LEVEL

Currently not used.

Default -1

Accepted [-inf; +inf]

Groups *Overall solver*, *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH

The linear dependency check is potentially computationally expensive.

Default 100

Accepted [-inf; +inf]

Groups *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH

The linear dependency check is potentially computationally expensive.

Default 100

Accepted [-inf; +inf]

Groups *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_USE

Controls whether the linear constraints are checked for linear dependencies.

Default *ON*

Accepted

- *ON*: Turns the linear dependency check on.
- *OFF*: Turns the linear dependency check off.

Groups *Presolve*

MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS
Controls the maximum number of reductions performed by the presolve. The value of the parameter is normally only changed in connection with debugging. A negative value implies that an infinite number of reductions are allowed.

Default -1

Accepted [-inf; +inf]

Groups *Overall solver*, *Presolve*

MSK_IPAR_PRESOLVE_USE
Controls whether the presolve is applied to a problem before it is optimized.

Default *FREE*

Accepted *OFF*, *ON*, *FREE*

Groups *Overall solver*, *Presolve*

MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER
Controls which optimizer that is used to find the optimal repair.

Default *FREE*

Accepted *FREE*, *INTPNT*, *CONIC*, *PRIMAL_SIMPLEX*, *DUAL_SIMPLEX*, *FREE_SIMPLEX*, *MIXED_INT*

Groups *Overall solver*

MSK_IPAR_READ_DATA_COMPRESSED
If this option is turned on,it is assumed that the data file is compressed.

Default *FREE*

Accepted *NONE*, *FREE*, *GZIP*

Groups *Data input/output*

MSK_IPAR_READ_DATA_FORMAT
Format of the data file to be read.

Default *EXTENSION*

Accepted *EXTENSION*, *MPS*, *LP*, *OP*, *XML*, *FREE_MPS*, *TASK*, *CB*, *JSON_TASK*

Groups *Data input/output*

MSK_IPAR_READ_DEBUG
Turns on additional debugging information when reading files.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_READ_KEEP_FREE_CON
Controls whether the free constraints are included in the problem.

Default *OFF*

Accepted

- *ON*: The free constraints are kept.

- *OFF*: The free constraints are discarded.

Groups *Data input/output*

MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU
If this option is turned on, **MOSEK** will drop variables that are defined for the first time in the bounds section.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_READ_LP_QUOTED_NAMES
If a name is in quotes when reading an LP file, the quotes will be removed.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_READ_MPS_FORMAT
Controls how strictly the MPS file reader interprets the MPS format.

Default *FREE*

Accepted *STRICT*, *RELAXED*, *FREE*, *CPLEX*

Groups *Data input/output*

MSK_IPAR_READ_MPS_WIDTH
Controls the maximal number of characters allowed in one line of the MPS file.

Default 1024

Accepted [80; +inf]

Groups *Data input/output*

MSK_IPAR_READ_TASK_IGNORE_PARAM
Controls whether **MOSEK** should ignore the parameter setting defined in the task file and use the default parameter setting instead.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_REMOVE_UNUSED_SOLUTIONS
Removes unsued solutions before the optimization is performed.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Overall system*

MSK_IPAR_SENSITIVITY_ALL
Not applicable.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Overall solver*

MSK_IPAR_SENSITIVITY_OPTIMIZER
Controls which optimizer is used for optimal partition sensitivity analysis.

Default *FREE_SIMPLEX*

Accepted *FREE*, *INTPNT*, *CONIC*, *PRIMAL_SIMPLEX*, *DUAL_SIMPLEX*, *FREE_SIMPLEX*, *MIXED_INT*

Groups *Overall solver*, *Simplex optimizer*

MSK_IPAR_SENSITIVITY_TYPE

Controls which type of sensitivity analysis is to be performed.

Default *BASIS*

Accepted *BASIS*, *OPTIMAL_PARTITION*

Groups *Overall solver*

MSK_IPAR_SIM_BASIS_FACTOR_USE

Controls whether an LU factorization of the basis is used in a hot-start. Forcing a refactorization sometimes improves the stability of the simplex optimizers, but in most cases there is a performance penalty.

Default *ON*

Accepted *ON*, *OFF*

Groups *Simplex optimizer*

MSK_IPAR_SIM_DEGEN

Controls how aggressively degeneration is handled.

Default *FREE*

Accepted *NONE*, *FREE*, *AGGRESSIVE*, *MODERATE*, *MINIMUM*

Groups *Simplex optimizer*

MSK_IPAR_SIM_DUAL_CRASH

Controls whether crashing is performed in the dual simplex optimizer.

If this parameter is set to x, then a crash will be performed if a basis consists of more than $(100-x)$ mod f_v entries, where f_v is the number of fixed variables.

Default 90

Accepted [0; +inf]

Groups *Dual simplex*

MSK_IPAR_SIM_DUAL_PHASEONE_METHOD

An experimental feature.

Default 0

Accepted [0; 10]

Groups *Simplex optimizer*

MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION

The dual simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the dual simplex optimizer first choose a subset of all the potential outgoing variables. Next, for some time it will choose the outgoing variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

Default 50

Accepted [0; 100]

Groups *Dual simplex*

`MSK_IPAR_SIM_DUAL_SELECTION`
Controls the choice of the incoming variable, known as the selection strategy, in the dual simplex optimizer.

Default *FREE*

Accepted *FREE*, *FULL*, *ASE*, *DEVEX*, *SE*, *PARTIAL*

Groups *Dual simplex*

`MSK_IPAR_SIM_EXPLOIT_DUPVEC`
Controls if the simplex optimizers are allowed to exploit duplicated columns.

Default *OFF*

Accepted *ON*, *OFF*, *FREE*

Groups *Simplex optimizer*

`MSK_IPAR_SIM_HOTSTART`
Controls the type of hot-start that the simplex optimizer perform.

Default *FREE*

Accepted *NONE*, *FREE*, *STATUS_KEYS*

Groups *Simplex optimizer*

`MSK_IPAR_SIM_HOTSTART_LU`
Determines if the simplex optimizer should exploit the initial factorization.

Default *ON*

Accepted

- *ON*: Factorization is reused if possible.
- *OFF*: Factorization is recomputed.

Groups *Simplex optimizer*

`MSK_IPAR_SIM_MAX_ITERATIONS`
Maximum number of iterations that can be used by a simplex optimizer.

Default 10000000

Accepted [0; +inf]

Groups *Simplex optimizer*, *Termination criteria*

`MSK_IPAR_SIM_MAX_NUM_SETBACKS`
Controls how many set-backs are allowed within a simplex optimizer. A set-back is an event where the optimizer moves in the wrong direction. This is impossible in theory but may happen due to numerical problems.

Default 250

Accepted [0; +inf]

Groups *Simplex optimizer*

`MSK_IPAR_SIM_NON_SINGULAR`
Controls if the simplex optimizer ensures a non-singular basis, if possible.

Default *ON*

Accepted *ON*, *OFF*

Groups *Simplex optimizer*

MSK_IPAR_SIM_PRIMAL_CRASH
Controls whether crashing is performed in the primal simplex optimizer.

In general, if a basis consists of more than (100-this parameter value)% fixed variables, then a crash will be performed.

Default 90

Accepted [0; +inf]

Groups *Primal simplex*

MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD
An experimental feature.

Default 0

Accepted [0; 10]

Groups *Simplex optimizer*

MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION
The primal simplex optimizer can use a so-called restricted selection/pricing strategy to chooses the outgoing variable. Hence, if restricted selection is applied, then the primal simplex optimizer first choose a subset of all the potential incoming variables. Next, for some time it will choose the incoming variable only among the subset. From time to time the subset is redefined.

A larger value of this parameter implies that the optimizer will be more aggressive in its restriction strategy, i.e. a value of 0 implies that the restriction strategy is not applied at all.

Default 50

Accepted [0; 100]

Groups *Primal simplex*

MSK_IPAR_SIM_PRIMAL_SELECTION
Controls the choice of the incoming variable, known as the selection strategy, in the primal simplex optimizer.

Default *FREE*

Accepted *FREE*, *FULL*, *ASE*, *DEVEX*, *SE*, *PARTIAL*

Groups *Primal simplex*

MSK_IPAR_SIM_REFACTOR_FREQ
Controls how frequent the basis is refactorized. The value 0 means that the optimizer determines the best point of refactorization.

It is strongly recommended NOT to change this parameter.

Default 0

Accepted [0; +inf]

Groups *Simplex optimizer*

MSK_IPAR_SIM_REFORMULATION
Controls if the simplex optimizers are allowed to reformulate the problem.

Default *OFF*

Accepted *ON*, *OFF*, *FREE*, *AGGRESSIVE*

Groups *Simplex optimizer*

MSK_IPAR_SIM_SAVE_LU
Controls if the LU factorization stored should be replaced with the LU factorization corresponding to the initial basis.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Simplex optimizer*

MSK_IPAR_SIM_SCALING

Controls how much effort is used in scaling the problem before a simplex optimizer is used.

Default *FREE*

Accepted *FREE*, *NONE*, *MODERATE*, *AGGRESSIVE*

Groups *Simplex optimizer*

MSK_IPAR_SIM_SCALING_METHOD

Controls how the problem is scaled before a simplex optimizer is used.

Default *POW2*

Accepted *POW2*, *FREE*

Groups *Simplex optimizer*

MSK_IPAR_SIM_SOLVE_FORM

Controls whether the primal or the dual problem is solved by the primal-/dual-simplex optimizer.

Default *FREE*

Accepted *FREE*, *PRIMAL*, *DUAL*

Groups *Simplex optimizer*

MSK_IPAR_SIM_STABILITY_PRIORITY

Controls how high priority the numerical stability should be given.

Default 50

Accepted [0; 100]

Groups *Simplex optimizer*

MSK_IPAR_SIM_SWITCH_OPTIMIZER

The simplex optimizer sometimes chooses to solve the dual problem instead of the primal problem. This implies that if you have chosen to use the dual simplex optimizer and the problem is dualized, then it actually makes sense to use the primal simplex optimizer instead. If this parameter is on and the problem is dualized and furthermore the simplex optimizer is chosen to be the primal (dual) one, then it is switched to the dual (primal).

Default *OFF*

Accepted *ON*, *OFF*

Groups *Simplex optimizer*

MSK_IPAR_SOL_FILTER_KEEP_BASIC

If turned on, then basic and super basic constraints and variables are written to the solution file independent of the filter setting.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Solution input/output*

MSK_IPAR_SOL_FILTER_KEEP_RANGED

If turned on, then ranged constraints and variables are written to the solution file independent of the filter setting.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Solution input/output*

MSK_IPAR_SOL_READ_NAME_WIDTH

When a solution is read by **MOSEK** and some constraint, variable or cone names contain blanks, then a maximum name width much be specified. A negative value implies that no name contain blanks.

Default -1

Accepted [-inf; +inf]

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_SOL_READ_WIDTH

Controls the maximal acceptable width of line in the solutions when read by **MOSEK**.

Default 1024

Accepted [80; +inf]

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_SOLUTION_CALLBACK

Indicates whether solution callbacks will be performed during the optimization.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Progress callback*, *Overall solver*

MSK_IPAR_TIMING_LEVEL

Controls the amount of timing performed inside **MOSEK**.

Default 1

Accepted [0; +inf]

Groups *Overall system*

MSK_IPAR_WRITE_BAS_CONSTRAINTS

Controls whether the constraint section is written to the basic solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_BAS_HEAD

Controls whether the header section is written to the basic solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_BAS_VARIABLES

Controls whether the variables section is written to the basic solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_DATA_COMPRESSED

Controls whether the data file is compressed while it is written. 0 means no compression while higher values mean more compression.

Default 0

Accepted [0; +inf]

Groups *Data input/output*

MSK_IPAR_WRITE_DATA_FORMAT
Controls the file format when writing task data to a file.

Default *EXTENSION*

Accepted *EXTENSION*, *MPS*, *LP*, *OP*, *XML*, *FREE_MPS*, *TASK*, *CB*, *JSON_TASK*

Groups *Data input/output*

MSK_IPAR_WRITE_DATA_PARAM
If this option is turned on the parameter settings are written to the data file as parameters.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_WRITE_FREE_CON
Controls whether the free constraints are written to the data file.

Default *ON*

Accepted

- *ON*: The free constraints are written.
- *OFF*: The free constraints are discarded.

Groups *Data input/output*

MSK_IPAR_WRITE_GENERIC_NAMES
Controls whether the generic names or user-defined names are used in the data file.

Default *OFF*

Accepted

- *ON*: Generic names are used.
- *OFF*: Generic names are not used.

Groups *Data input/output*

MSK_IPAR_WRITE_GENERIC_NAMES_IO
Index origin used in generic names.

Default 1

Accepted [0; +inf]

Groups *Data input/output*

MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS
Controls if the writer ignores incompatible problem items when writing files.

Default *OFF*

Accepted

- *ON*: Ignore items that cannot be written to the current output file format.
- *OFF*: Produce an error if the problem contains items that cannot the written to the current output file format.

Groups *Data input/output*

MSK_IPAR_WRITE_INT_CONSTRAINTS
Controls whether the constraint section is written to the integer solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_INT_HEAD
Controls whether the header section is written to the integer solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_INT_VARIABLES
Controls whether the variables section is written to the integer solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_LP_FULL_OBJ
Write all variables, including the ones with 0-coefficients, in the objective.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_WRITE_LP_LINE_WIDTH
Maximum width of line in an LP file written by **MOSEK**.

Default 80

Accepted [40; +inf]

Groups *Data input/output*

MSK_IPAR_WRITE_LP_QUOTED_NAMES
If this option is turned on, then **MOSEK** will quote invalid LP names when writing an LP file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_WRITE_LP_STRICT_FORMAT
Controls whether LP output files satisfy the LP format strictly.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_WRITE_LP_TERMS_PER_LINE
Maximum number of terms on a single line in an LP file written by **MOSEK**. 0 means unlimited.

Default 10

Accepted [0; +inf]

Groups *Data input/output*

MSK_IPAR_WRITE_MPS_FORMAT
Controls in which format the MPS is written.

Default *FREE*

Accepted *STRICT*, *RELAXED*, *FREE*, *CPLEX*

Groups *Data input/output*

MSK_IPAR_WRITE_MPS_INT

Controls if marker records are written to the MPS file to indicate whether variables are integer restricted.

Default *ON*

Accepted

- *ON*: Marker records are written.
- *OFF*: Marker records are not written.

Groups *Data input/output*

MSK_IPAR_WRITE_PRECISION

Controls the precision with which `double` numbers are printed in the MPS data file. In general it is not worthwhile to use a value higher than 15.

Default 15

Accepted [0; +inf]

Groups *Data input/output*

MSK_IPAR_WRITE_SOL_BARVARIABLES

Controls whether the symmetric matrix variables section is written to the solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_SOL_CONSTRAINTS

Controls whether the constraint section is written to the solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_SOL_HEAD

Controls whether the header section is written to the solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES

Even if the names are invalid MPS names, then they are employed when writing the solution file.

Default *OFF*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_SOL_VARIABLES

Controls whether the variables section is written to the solution file.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*, *Solution input/output*

MSK_IPAR_WRITE_TASK_INC_SOL

Controls whether the solutions are stored in the task file too.

Default *ON*

Accepted *ON*, *OFF*

Groups *Data input/output*

MSK_IPAR_WRITE_XML_MODE
Controls if linear coefficients should be written by row or column when writing in the XML file format.

Default *ROW*

Accepted *ROW*, *COL*

Groups *Data input/output*

11.3.3 String parameters

MSK_SPAR_BAS_SOL_FILE_NAME
Name of the `bas` solution file.

Accepted Any valid file name.

Groups *Data input/output*, *Solution input/output*

MSK_SPAR_DATA_FILE_NAME
Data are read and written to this file.

Accepted Any valid file name.

Groups *Data input/output*

MSK_SPAR_DEBUG_FILE_NAME
MOSEK debug file.

Accepted Any valid file name.

Groups *Data input/output*

MSK_SPAR_INT_SOL_FILE_NAME
Name of the `int` solution file.

Accepted Any valid file name.

Groups *Data input/output*, *Solution input/output*

MSK_SPAR_ITR_SOL_FILE_NAME
Name of the `itr` solution file.

Accepted Any valid file name.

Groups *Data input/output*, *Solution input/output*

MSK_SPAR_MIO_DEBUG_STRING
For internal debugging purposes.

Accepted Any valid string.

Groups *Data input/output*

MSK_SPAR_PARAM_COMMENT_SIGN
Only the first character in this string is used. It is considered as a start of comment sign in the **MOSEK** parameter file. Spaces are ignored in the string.

Default

%%

Accepted Any valid string.

Groups *Data input/output*

`MSK_SPAR_PARAM_READ_FILE_NAME`

Modifications to the parameter database is read from this file.

Accepted Any valid file name.

Groups *Data input/output*

`MSK_SPAR_PARAM_WRITE_FILE_NAME`

The parameter database is written to this file.

Accepted Any valid file name.

Groups *Data input/output*

`MSK_SPAR_READ_MPS_BOU_NAME`

Name of the BOUNDS vector used. An empty name means that the first BOUNDS vector is used.

Accepted Any valid MPS name.

Groups *Data input/output*

`MSK_SPAR_READ_MPS_OBJ_NAME`

Name of the free constraint used as objective function. An empty name means that the first constraint is used as objective function.

Accepted Any valid MPS name.

Groups *Data input/output*

`MSK_SPAR_READ_MPS_RAN_NAME`

Name of the RANGE vector used. An empty name means that the first RANGE vector is used.

Accepted Any valid MPS name.

Groups *Data input/output*

`MSK_SPAR_READ_MPS_RHS_NAME`

Name of the RHS used. An empty name means that the first RHS vector is used.

Accepted Any valid MPS name.

Groups *Data input/output*

`MSK_SPAR_REMOTE_ACCESS_TOKEN`

An access token used to submit tasks to a remote **MOSEK** server. An access token is a random 32-byte string encoded in base64, i.e. it is a 44 character ASCII string.

Accepted Any valid string.

Groups *Overall system*

`MSK_SPAR_SENSITIVITY_FILE_NAME`

If defined, **MOSEK** reads this file as a sensitivity analysis data file specifying the type of analysis to be done.

Accepted Any valid string.

Groups *Data input/output*

`MSK_SPAR_SENSITIVITY_RES_FILE_NAME`

Accepted Any valid string.

Groups *Data input/output*

`MSK_SPAR_SOL_FILTER_XC_LOW`

A filter used to determine which constraints should be listed in the solution file. A value of 0.5 means that all constraints having `xc[i]>0.5` should be listed, whereas `+0.5` means that all constraints having `xc[i]>=blc[i]+0.5` should be listed. An empty filter means that no filter is applied.

Accepted Any valid filter.

Groups *Data input/output*, *Solution input/output*

MSK_SPAR_SOL_FILTER_XC_UPR
A filter used to determine which constraints should be listed in the solution file. A value of `0.5` means that all constraints having `xc[i]<0.5` should be listed, whereas `-0.5` means all constraints having `xc[i]<=buc[i]-0.5` should be listed. An empty filter means that no filter is applied.

Accepted Any valid filter.

Groups *Data input/output*, *Solution input/output*

MSK_SPAR_SOL_FILTER_XX_LOW
A filter used to determine which variables should be listed in the solution file. A value of "0.5" means that all constraints having `xx[j]>=0.5` should be listed, whereas "+0.5" means that all constraints having `xx[j]>=blx[j]+0.5` should be listed. An empty filter means no filter is applied.

Accepted Any valid filter.

Groups *Data input/output*, *Solution input/output*

MSK_SPAR_SOL_FILTER_XX_UPR
A filter used to determine which variables should be listed in the solution file. A value of "0.5" means that all constraints having `xx[j]<0.5` should be printed, whereas "-0.5" means all constraints having `xx[j]<=bux[j]-0.5` should be listed. An empty filter means no filter is applied.

Accepted Any valid file name.

Groups *Data input/output*, *Solution input/output*

MSK_SPAR_STAT_FILE_NAME
Statistics file name.

Accepted Any valid file name.

Groups *Data input/output*

MSK_SPAR_STAT_KEY
Key used when writing the summary file.

Accepted Any valid string.

Groups *Data input/output*

MSK_SPAR_STAT_NAME
Name used when writing the statistics file.

Accepted Any valid XML string.

Groups *Data input/output*

MSK_SPAR_WRITE_LP_GEN_VAR_NAME
Sometimes when an LP file is written additional variables must be inserted. They will have the prefix denoted by this parameter.

Default xmskgen

Accepted Any valid string.

Groups *Data input/output*

11.4 Response codes

- *Termination*
- *Warnings*
- *Errors*

11.4.1 Termination

MSK_RES_OK
: No error occurred.

MSK_RES_TRM_MAX_ITERATIONS
: The optimizer terminated at the maximum number of iterations.

MSK_RES_TRM_MAX_TIME
: The optimizer terminated at the maximum amount of time.

MSK_RES_TRM_OBJECTIVE_RANGE
: The optimizer terminated with an objective value outside the objective range.

MSK_RES_TRM_MIO_NEAR_REL_GAP
: The mixed-integer optimizer terminated as the delayed near optimal relative gap tolerance was satisfied.

MSK_RES_TRM_MIO_NEAR_ABS_GAP
: The mixed-integer optimizer terminated as the delayed near optimal absolute gap tolerance was satisfied.

MSK_RES_TRM_MIO_NUM_RELAXS
: The mixed-integer optimizer terminated as the maximum number of relaxations was reached.

MSK_RES_TRM_MIO_NUM_BRANCHES
: The mixed-integer optimizer terminated as the maximum number of branches was reached.

MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS
: The mixed-integer optimizer terminated as the maximum number of feasible solutions was reached.

MSK_RES_TRM_STALL
: The optimizer is terminated due to slow progress.

Stalling means that numerical problems prevent the optimizer from making reasonable progress and that it make no sense to continue. In many cases this happens if the problem is badly scaled or otherwise ill-conditioned. There is no guarantee that the solution will be (near) feasible or near optimal. However, often stalling happens near the optimum, and the returned solution may be of good quality. Therefore, it is recommended to check the status of then solution. If the solution near optimal the solution is most likely good enough for most practical purposes.

Please note that if a linear optimization problem is solved using the interior-point optimizer with basis identification turned on, the returned basic solution likely to have high accuracy, even though the optimizer stalled.

Some common causes of stalling are a) badly scaled models, b) near feasible or near infeasible problems and c) a non-convex problems. Case c) is only relevant for general non-linear problems. It is not possible in general for **MOSEK** to check if a specific problems is convex since such a check would be NP hard in itself. This implies that care should be taken when solving problems involving general user defined functions.

MSK_RES_TRM_USER_CALLBACK
: The optimizer terminated due to the return of the user-defined callback function.

MSK_RES_TRM_MAX_NUM_SETBACKS
: The optimizer terminated as the maximum number of set-backs was reached. This indicates serious numerical problems and a possibly badly formulated problem.

MSK_RES_TRM_NUMERICAL_PROBLEM
: The optimizer terminated due to numerical problems.

MSK_RES_TRM_INTERNAL
: The optimizer terminated due to some internal reason. Please contact **MOSEK** support.

MSK_RES_TRM_INTERNAL_STOP
: The optimizer terminated for internal reasons. Please contact **MOSEK** support.

11.4.2 Warnings

MSK_RES_WRN_OPEN_PARAM_FILE
The parameter file could not be opened.

MSK_RES_WRN_LARGE_BOUND
A numerically large bound value is specified.

MSK_RES_WRN_LARGE_LO_BOUND
A numerically large lower bound value is specified.

MSK_RES_WRN_LARGE_UP_BOUND
A numerically large upper bound value is specified.

MSK_RES_WRN_LARGE_CON_FX
An equality constraint is fixed to a numerically large value. This can cause numerical problems.

MSK_RES_WRN_LARGE_CJ
A numerically large value is specified for one c_j.

MSK_RES_WRN_LARGE_AIJ
A numerically large value is specified for an $a_{i,j}$ element in A. The parameter *MSK_DPAR_DATA_TOL_AIJ_LARGE* controls when an $a_{i,j}$ is considered large.

MSK_RES_WRN_ZERO_AIJ
One or more zero elements are specified in A.

MSK_RES_WRN_NAME_MAX_LEN
A name is longer than the buffer that is supposed to hold it.

MSK_RES_WRN_SPAR_MAX_LEN
A value for a string parameter is longer than the buffer that is supposed to hold it.

MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR
An RHS vector is split into several nonadjacent parts in an MPS file.

MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR
A RANGE vector is split into several nonadjacent parts in an MPS file.

MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR
A BOUNDS vector is split into several nonadjacent parts in an MPS file.

MSK_RES_WRN_LP_OLD_QUAD_FORMAT
Missing '/2' after quadratic expressions in bound or objective.

MSK_RES_WRN_LP_DROP_VARIABLE
Ignored a variable because the variable was not previously defined. Usually this implies that a variable appears in the bound section but not in the objective or the constraints.

MSK_RES_WRN_NZ_IN_UPR_TRI
Non-zero elements specified in the upper triangle of a matrix were ignored.

MSK_RES_WRN_DROPPED_NZ_QOBJ
One or more non-zero elements were dropped in the Q matrix in the objective.

MSK_RES_WRN_IGNORE_INTEGER
Ignored integer constraints.

MSK_RES_WRN_NO_GLOBAL_OPTIMIZER
No global optimizer is available.

MSK_RES_WRN_MIO_INFEASIBLE_FINAL
The final mixed-integer problem with all the integer variables fixed at their optimal values is infeasible.

MSK_RES_WRN_SOL_FILTER
Invalid solution filter is specified.

MSK_RES_WRN_UNDEF_SOL_FILE_NAME
Undefined name occurred in a solution.

MSK_RES_WRN_SOL_FILE_IGNORED_CON
One or more lines in the constraint section were ignored when reading a solution file.

MSK_RES_WRN_SOL_FILE_IGNORED_VAR
One or more lines in the variable section were ignored when reading a solution file.

MSK_RES_WRN_TOO_FEW_BASIS_VARS
An incomplete basis has been specified. Too few basis variables are specified.

MSK_RES_WRN_TOO_MANY_BASIS_VARS
A basis with too many variables has been specified.

MSK_RES_WRN_NO_NONLINEAR_FUNCTION_WRITE
The problem contains a general nonlinear function in either the objective or the constraints. Such a nonlinear function cannot be written to a disk file. Note that quadratic terms when inputted explicitly can be written to disk.

MSK_RES_WRN_LICENSE_EXPIRE
The license expires.

MSK_RES_WRN_LICENSE_SERVER
The license server is not responding.

MSK_RES_WRN_EMPTY_NAME
A variable or constraint name is empty. The output file may be invalid.

MSK_RES_WRN_USING_GENERIC_NAMES
Generic names are used because a name is not valid. For instance when writing an LP file the names must not contain blanks or start with a digit.

MSK_RES_WRN_LICENSE_FEATURE_EXPIRE
The license expires.

MSK_RES_WRN_PARAM_NAME_DOU
The parameter name is not recognized as a double parameter.

MSK_RES_WRN_PARAM_NAME_INT
The parameter name is not recognized as a integer parameter.

MSK_RES_WRN_PARAM_NAME_STR
The parameter name is not recognized as a string parameter.

MSK_RES_WRN_PARAM_STR_VALUE
The string is not recognized as a symbolic value for the parameter.

MSK_RES_WRN_PARAM_IGNORED_CMIO
A parameter was ignored by the conic mixed integer optimizer.

MSK_RES_WRN_ZEROS_IN_SPARSE_ROW
One or more (near) zero elements are specified in a sparse row of a matrix. Since, it is redundant to specify zero elements then it may indicate an error.

MSK_RES_WRN_ZEROS_IN_SPARSE_COL
One or more (near) zero elements are specified in a sparse column of a matrix. It is redundant to specify zero elements. Hence, it may indicate an error.

MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK
The linear dependency check(s) is incomplete. Normally this is not an important warning unless the optimization problem has been formulated with linear dependencies. Linear dependencies may prevent **MOSEK** from solving the problem.

MSK_RES_WRN_ELIMINATOR_SPACE
The eliminator is skipped at least once due to lack of space.

MSK_RES_WRN_PRESOLVE_OUTOFSPACE
The presolve is incomplete due to lack of space.

MSK_RES_WRN_WRITE_CHANGED_NAMES
Some names were changed because they were invalid for the output file format.

MSK_RES_WRN_WRITE_DISCARDED_CFIX
The fixed objective term could not be converted to a variable and was discarded in the output file.

MSK_RES_WRN_CONSTRUCT_SOLUTION_INFEAS
After fixing the integer variables at the suggested values then the problem is infeasible.

MSK_RES_WRN_CONSTRUCT_INVALID_SOL_ITG
The initial value for one or more of the integer variables is not feasible.

MSK_RES_WRN_CONSTRUCT_NO_SOL_ITG
The construct solution requires an integer solution.

MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES
Two constraint names are identical.

MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES
Two variable names are identical.

MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES
Two barvariable names are identical.

MSK_RES_WRN_DUPLICATE_CONE_NAMES
Two cone names are identical.

MSK_RES_WRN_ANA_LARGE_BOUNDS
This warning is issued by the problem analyzer, if one or more constraint or variable bounds are very large. One should consider omitting these bounds entirely by setting them to +inf or -inf.

MSK_RES_WRN_ANA_C_ZERO
This warning is issued by the problem analyzer, if the coefficients in the linear part of the objective are all zero.

MSK_RES_WRN_ANA_EMPTY_COLS
This warning is issued by the problem analyzer, if columns, in which all coefficients are zero, are found.

MSK_RES_WRN_ANA_CLOSE_BOUNDS
This warning is issued by problem analyzer, if ranged constraints or variables with very close upper and lower bounds are detected. One should consider treating such constraints as equalities and such variables as constants.

MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS
This warning is issued by the problem analyzer if a constraint is bound nearly integral.

MSK_RES_WRN_QUAD_CONES_WITH_ROOT_FIXED_AT_ZERO
For at least one quadratic cone the root is fixed at (nearly) zero. This may cause problems such as a very large dual solution. Therefore, it is recommended to remove such cones before optimizing the problems, or to fix all the variables in the cone to 0.

MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO
For at least one rotated quadratic cone at least one of the root variables are fixed at (nearly) zero. This may cause problems such as a very large dual solution. Therefore, it is recommended to remove such cones before optimizing the problems, or to fix all the variables in the cone to 0.

MSK_RES_WRN_NO_DUALIZER
No automatic dualizer is available for the specified problem. The primal problem is solved.

MSK_RES_WRN_SYM_MAT_LARGE
A numerically large value is specified for an $e_{i,j}$ element in E. The parameter *MSK_DPAR_DATA_SYM_MAT_TOL_LARGE* controls when an $e_{i,j}$ is considered large.

11.4.3 Errors

`MSK_RES_ERR_LICENSE`
Invalid license.

`MSK_RES_ERR_LICENSE_EXPIRED`
The license has expired.

`MSK_RES_ERR_LICENSE_VERSION`
The license is valid for another version of **MOSEK**.

`MSK_RES_ERR_SIZE_LICENSE`
The problem is bigger than the license.

`MSK_RES_ERR_PROB_LICENSE`
The software is not licensed to solve the problem.

`MSK_RES_ERR_FILE_LICENSE`
Invalid license file.

`MSK_RES_ERR_MISSING_LICENSE_FILE`
MOSEK cannot license file or a token server. See the **MOSEK** installation manual for details.

`MSK_RES_ERR_SIZE_LICENSE_CON`
The problem has too many constraints to be solved with the available license.

`MSK_RES_ERR_SIZE_LICENSE_VAR`
The problem has too many variables to be solved with the available license.

`MSK_RES_ERR_SIZE_LICENSE_INTVAR`
The problem contains too many integer variables to be solved with the available license.

`MSK_RES_ERR_OPTIMIZER_LICENSE`
The optimizer required is not licensed.

`MSK_RES_ERR_FLEXLM`
The FLEXlm license manager reported an error.

`MSK_RES_ERR_LICENSE_SERVER`
The license server is not responding.

`MSK_RES_ERR_LICENSE_MAX`
Maximum number of licenses is reached.

`MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON`
The MOSEKLM license manager daemon is not up and running.

`MSK_RES_ERR_LICENSE_FEATURE`
A requested feature is not available in the license file(s). Most likely due to an incorrect license system setup.

`MSK_RES_ERR_PLATFORM_NOT_LICENSED`
A requested license feature is not available for the required platform.

`MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE`
The license system cannot allocate the memory required.

`MSK_RES_ERR_LICENSE_CANNOT_CONNECT`
MOSEK cannot connect to the license server. Most likely the license server is not up and running.

`MSK_RES_ERR_LICENSE_INVALID_HOSTID`
The host ID specified in the license file does not match the host ID of the computer.

`MSK_RES_ERR_LICENSE_SERVER_VERSION`
The version specified in the checkout request is greater than the highest version number the daemon supports.

`MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT`
The license server does not support the requested feature. Possible reasons for this error include:

- The feature has expired.
- The feature's start date is later than today's date.
- The version requested is higher than feature's the highest supported version.
- A corrupted license file.

Try restarting the license and inspect the license server debug file, usually called `lmgrd.log`.

`MSK_RES_ERR_LICENSE_NO_SERVER_LINE`
There is no `SERVER` line in the license file. All non-zero license count features need at least one `SERVER` line.

`MSK_RES_ERR_OPEN_DL`
A dynamic link library could not be opened.

`MSK_RES_ERR_OLDER_DLL`
The dynamic link library is older than the specified version.

`MSK_RES_ERR_NEWER_DLL`
The dynamic link library is newer than the specified version.

`MSK_RES_ERR_LINK_FILE_DLL`
A file cannot be linked to a stream in the DLL version.

`MSK_RES_ERR_THREAD_MUTEX_INIT`
Could not initialize a mutex.

`MSK_RES_ERR_THREAD_MUTEX_LOCK`
Could not lock a mutex.

`MSK_RES_ERR_THREAD_MUTEX_UNLOCK`
Could not unlock a mutex.

`MSK_RES_ERR_THREAD_CREATE`
Could not create a thread. This error may occur if a large number of environments are created and not deleted again. In any case it is a good practice to minimize the number of environments created.

`MSK_RES_ERR_THREAD_COND_INIT`
Could not initialize a condition.

`MSK_RES_ERR_UNKNOWN`
Unknown error.

`MSK_RES_ERR_SPACE`
Out of space.

`MSK_RES_ERR_FILE_OPEN`
Error while opening a file.

`MSK_RES_ERR_FILE_READ`
File read error.

`MSK_RES_ERR_FILE_WRITE`
File write error.

`MSK_RES_ERR_DATA_FILE_EXT`
The data file format cannot be determined from the file name.

`MSK_RES_ERR_INVALID_FILE_NAME`
An invalid file name has been specified.

`MSK_RES_ERR_INVALID_SOL_FILE_NAME`
An invalid file name has been specified.

`MSK_RES_ERR_END_OF_FILE`
End of file reached.

MSK_RES_ERR_NULL_ENV
: `env` is a NULL pointer.

MSK_RES_ERR_NULL_TASK
: `task` is a NULL pointer.

MSK_RES_ERR_INVALID_STREAM
: An invalid stream is referenced.

MSK_RES_ERR_NO_INIT_ENV
: `env` is not initialized.

MSK_RES_ERR_INVALID_TASK
: The `task` is invalid.

MSK_RES_ERR_NULL_POINTER
: An argument to a function is unexpectedly a NULL pointer.

MSK_RES_ERR_LIVING_TASKS
: All tasks associated with an enviroment must be deleted before the environment is deleted. There are still some undeleted tasks.

MSK_RES_ERR_BLANK_NAME
: An all blank name has been specified.

MSK_RES_ERR_DUP_NAME
: The same name was used multiple times for the same problem item type.

MSK_RES_ERR_INVALID_OBJ_NAME
: An invalid objective name is specified.

MSK_RES_ERR_INVALID_CON_NAME
: An invalid constraint name is used.

MSK_RES_ERR_INVALID_VAR_NAME
: An invalid variable name is used.

MSK_RES_ERR_INVALID_CONE_NAME
: An invalid cone name is used.

MSK_RES_ERR_INVALID_BARVAR_NAME
: An invalid symmetric matrix variable name is used.

MSK_RES_ERR_SPACE_LEAKING
: **MOSEK** is leaking memory. This can be due to either an incorrect use of **MOSEK** or a bug.

MSK_RES_ERR_SPACE_NO_INFO
: No available information about the space usage.

MSK_RES_ERR_READ_FORMAT
: The specified format cannot be read.

MSK_RES_ERR_MPS_FILE
: An error occurred while reading an MPS file.

MSK_RES_ERR_MPS_INV_FIELD
: A field in the MPS file is invalid. Probably it is too wide.

MSK_RES_ERR_MPS_INV_MARKER
: An invalid marker has been specified in the MPS file.

MSK_RES_ERR_MPS_NULL_CON_NAME
: An empty constraint name is used in an MPS file.

MSK_RES_ERR_MPS_NULL_VAR_NAME
: An empty variable name is used in an MPS file.

MSK_RES_ERR_MPS_UNDEF_CON_NAME
: An undefined constraint name occurred in an MPS file.

MSK_RES_ERR_MPS_UNDEF_VAR_NAME
    An undefined variable name occurred in an MPS file.

MSK_RES_ERR_MPS_INV_CON_KEY
    An invalid constraint key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_BOUND_KEY
    An invalid bound key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_SEC_NAME
    An invalid section name occurred in an MPS file.

MSK_RES_ERR_MPS_NO_OBJECTIVE
    No objective is defined in an MPS file.

MSK_RES_ERR_MPS_SPLITTED_VAR
    All elements in a column of the A matrix must be specified consecutively. Hence, it is illegal to specify non-zero elements in A for variable 1, then for variable 2 and then variable 1 again.

MSK_RES_ERR_MPS_MUL_CON_NAME
    A constraint name was specified multiple times in the `ROWS` section.

MSK_RES_ERR_MPS_MUL_QSEC
    Multiple `QSECTION`s are specified for a constraint in the MPS data file.

MSK_RES_ERR_MPS_MUL_QOBJ
    The Q term in the objective is specified multiple times in the MPS data file.

MSK_RES_ERR_MPS_INV_SEC_ORDER
    The sections in the MPS data file are not in the correct order.

MSK_RES_ERR_MPS_MUL_CSEC
    Multiple `CSECTION`s are given the same name.

MSK_RES_ERR_MPS_CONE_TYPE
    Invalid cone type specified in a `CSECTION`.

MSK_RES_ERR_MPS_CONE_OVERLAP
    A variable is specified to be a member of several cones.

MSK_RES_ERR_MPS_CONE_REPEAT
    A variable is repeated within the `CSECTION`.

MSK_RES_ERR_MPS_NON_SYMMETRIC_Q
    A non symmetric matrice has been speciefied.

MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT
    Duplicate elements is specfied in a Q matrix.

MSK_RES_ERR_MPS_INVALID_OBJSENSE
    An invalid objective sense is specified.

MSK_RES_ERR_MPS_TAB_IN_FIELD2
    A tab char occurred in field 2.

MSK_RES_ERR_MPS_TAB_IN_FIELD3
    A tab char occurred in field 3.

MSK_RES_ERR_MPS_TAB_IN_FIELD5
    A tab char occurred in field 5.

MSK_RES_ERR_MPS_INVALID_OBJ_NAME
    An invalid objective name is specified.

MSK_RES_ERR_LP_INCOMPATIBLE
    The problem cannot be written to an LP formatted file.

MSK_RES_ERR_LP_EMPTY
    The problem cannot be written to an LP formatted file.

`MSK_RES_ERR_LP_DUP_SLACK_NAME`
: The name of the slack variable added to a ranged constraint already exists.

`MSK_RES_ERR_WRITE_MPS_INVALID_NAME`
: An invalid name is created while writing an MPS file. Usually this will make the MPS file unreadable.

`MSK_RES_ERR_LP_INVALID_VAR_NAME`
: A variable name is invalid when used in an LP formatted file.

`MSK_RES_ERR_LP_FREE_CONSTRAINT`
: Free constraints cannot be written in LP file format.

`MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME`
: Empty variable names cannot be written to OPF files.

`MSK_RES_ERR_LP_FILE_FORMAT`
: Syntax error in an LP file.

`MSK_RES_ERR_WRITE_LP_FORMAT`
: Problem cannot be written as an LP file.

`MSK_RES_ERR_READ_LP_MISSING_END_TAG`
: Syntax error in LP file. Possibly missing End tag.

`MSK_RES_ERR_LP_FORMAT`
: Syntax error in an LP file.

`MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME`
: An auto-generated name is not unique.

`MSK_RES_ERR_READ_LP_NONEXISTING_NAME`
: A variable never occurred in objective or constraints.

`MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM`
: The problem contains cones that cannot be written to an LP formatted file.

`MSK_RES_ERR_LP_WRITE_GECO_PROBLEM`
: The problem contains general convex terms that cannot be written to an LP formatted file.

`MSK_RES_ERR_WRITING_FILE`
: An error occurred while writing file

`MSK_RES_ERR_OPF_FORMAT`
: Syntax error in an OPF file

`MSK_RES_ERR_OPF_NEW_VARIABLE`
: Introducing new variables is now allowed. When a `[variables]` section is present, it is not allowed to introduce new variables later in the problem.

`MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE`
: An invalid name occurred in a solution file.

`MSK_RES_ERR_LP_INVALID_CON_NAME`
: A constraint name is invalid when used in an LP formatted file.

`MSK_RES_ERR_OPF_PREMATURE_EOF`
: Premature end of file in an OPF file.

`MSK_RES_ERR_JSON_SYNTAX`
: Syntax error in an JSON data

`MSK_RES_ERR_JSON_STRING`
: Error in JSON string.

`MSK_RES_ERR_JSON_NUMBER_OVERFLOW`
: Invalid number entry - wrong type or value overflow.

MSK_RES_ERR_JSON_FORMAT
Error in an JSON Task file

MSK_RES_ERR_JSON_DATA
Inconsistent data in JSON Task file

MSK_RES_ERR_JSON_MISSING_DATA
Missing data section in JSON task file.

MSK_RES_ERR_ARGUMENT_LENNEQ
Incorrect length of arguments.

MSK_RES_ERR_ARGUMENT_TYPE
Incorrect argument type.

MSK_RES_ERR_NR_ARGUMENTS
Incorrect number of function arguments.

MSK_RES_ERR_IN_ARGUMENT
A function argument is incorrect.

MSK_RES_ERR_ARGUMENT_DIMENSION
A function argument is of incorrect dimension.

MSK_RES_ERR_INDEX_IS_TOO_SMALL
An index in an argument is too small.

MSK_RES_ERR_INDEX_IS_TOO_LARGE
An index in an argument is too large.

MSK_RES_ERR_PARAM_NAME
The parameter name is not correct.

MSK_RES_ERR_PARAM_NAME_DOU
The parameter name is not correct for a double parameter.

MSK_RES_ERR_PARAM_NAME_INT
The parameter name is not correct for an integer parameter.

MSK_RES_ERR_PARAM_NAME_STR
The parameter name is not correct for a string parameter.

MSK_RES_ERR_PARAM_INDEX
Parameter index is out of range.

MSK_RES_ERR_PARAM_IS_TOO_LARGE
The parameter value is too large.

MSK_RES_ERR_PARAM_IS_TOO_SMALL
The parameter value is too small.

MSK_RES_ERR_PARAM_VALUE_STR
The parameter value string is incorrect.

MSK_RES_ERR_PARAM_TYPE
The parameter type is invalid.

MSK_RES_ERR_INF_DOU_INDEX
A double information index is out of range for the specified type.

MSK_RES_ERR_INF_INT_INDEX
An integer information index is out of range for the specified type.

MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL
An index in an array argument is too small.

MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE
An index in an array argument is too large.

`MSK_RES_ERR_INF_LINT_INDEX`
A long integer information index is out of range for the specified type.

`MSK_RES_ERR_ARG_IS_TOO_SMALL`
The value of a argument is too small.

`MSK_RES_ERR_ARG_IS_TOO_LARGE`
The value of a argument is too small.

`MSK_RES_ERR_INVALID_WHICHSOL`
`whichsol` is invalid.

`MSK_RES_ERR_INF_DOU_NAME`
A double information name is invalid.

`MSK_RES_ERR_INF_INT_NAME`
An integer information name is invalid.

`MSK_RES_ERR_INF_TYPE`
The information type is invalid.

`MSK_RES_ERR_INF_LINT_NAME`
A long integer information name is invalid.

`MSK_RES_ERR_INDEX`
An index is out of range.

`MSK_RES_ERR_WHICHSOL`
The solution defined by `whichsol` does not exists.

`MSK_RES_ERR_SOLITEM`
The solution item number `solitem` is invalid. Please note that *`MSK_SOL_ITEM_SNX`* is invalid for the basic solution.

`MSK_RES_ERR_WHICHITEM_NOT_ALLOWED`
whichitem is unacceptable.

`MSK_RES_ERR_MAXNUMCON`
The maximum number of constraints specified is smaller than the number of constraints in the task.

`MSK_RES_ERR_MAXNUMVAR`
The maximum number of variables specified is smaller than the number of variables in the task.

`MSK_RES_ERR_MAXNUMBARVAR`
The maximum number of semidefinite variables specified is smaller than the number of semidefinite variables in the task.

`MSK_RES_ERR_MAXNUMQNZ`
The maximum number of non-zeros specified for the Q matrices is smaller than the number of non-zeros in the current Q matrices.

`MSK_RES_ERR_TOO_SMALL_MAX_NUM_NZ`
The maximum number of non-zeros specified is too small.

`MSK_RES_ERR_INVALID_IDX`
A specified index is invalid.

`MSK_RES_ERR_INVALID_MAX_NUM`
A specified index is invalid.

`MSK_RES_ERR_NUMCONLIM`
Maximum number of constraints limit is exceeded.

`MSK_RES_ERR_NUMVARLIM`
Maximum number of variables limit is exceeded.

MSK_RES_ERR_TOO_SMALL_MAXNUMANZ
The maximum number of non-zeros specified for A is smaller than the number of non-zeros in the current A.

MSK_RES_ERR_INV_APTRE
aptre[j] is strictly smaller than aptrb[j] for some j.

MSK_RES_ERR_MUL_A_ELEMENT
An element in A is defined multiple times.

MSK_RES_ERR_INV_BK
Invalid bound key.

MSK_RES_ERR_INV_BKC
Invalid bound key is specified for a constraint.

MSK_RES_ERR_INV_BKX
An invalid bound key is specified for a variable.

MSK_RES_ERR_INV_VAR_TYPE
An invalid variable type is specified for a variable.

MSK_RES_ERR_SOLVER_PROBTYPE
Problem type does not match the chosen optimizer.

MSK_RES_ERR_OBJECTIVE_RANGE
Empty objective range.

MSK_RES_ERR_FIRST
Invalid first.

MSK_RES_ERR_LAST
Invalid index last. A given index was out of expected range.

MSK_RES_ERR_NEGATIVE_SURPLUS
Negative surplus.

MSK_RES_ERR_NEGATIVE_APPEND
Cannot append a negative number.

MSK_RES_ERR_UNDEF_SOLUTION
MOSEK has the following solution types:

- an interior-point solution,
- an basic solution,
- and an integer solution.

Each optimizer may set one or more of these solutions; e.g by default a successful optimization with the interior-point optimizer defines the interior-point solution, and, for linear problems, also the basic solution. This error occurs when asking for a solution or for information about a solution that is not defined.

MSK_RES_ERR_BASIS
An invalid basis is specified. Either too many or too few basis variables are specified.

MSK_RES_ERR_INV_SKC
Invalid value in skc.

MSK_RES_ERR_INV_SKX
Invalid value in skx.

MSK_RES_ERR_INV_SKN
Invalid value in skn.

MSK_RES_ERR_INV_SK_STR
Invalid status key string encountered.

MSK_RES_ERR_INV_SK
: Invalid status key code.

MSK_RES_ERR_INV_CONE_TYPE_STR
: Invalid cone type string encountered.

MSK_RES_ERR_INV_CONE_TYPE
: Invalid cone type code is encountered.

MSK_RES_ERR_INVALID_SURPLUS
: Invalid surplus.

MSK_RES_ERR_INV_NAME_ITEM
: An invalid name item code is used.

MSK_RES_ERR_PRO_ITEM
: An invalid problem is used.

MSK_RES_ERR_INVALID_FORMAT_TYPE
: Invalid format type.

MSK_RES_ERR_FIRSTI
: Invalid `firsti`.

MSK_RES_ERR_LASTI
: Invalid `lasti`.

MSK_RES_ERR_FIRSTJ
: Invalid `firstj`.

MSK_RES_ERR_LASTJ
: Invalid `lastj`.

MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL
: An maximum length that is too small has been specified.

MSK_RES_ERR_NONLINEAR_EQUALITY
: The model contains a nonlinear equality which defines a nonconvex set.

MSK_RES_ERR_NONCONVEX
: The optimization problem is nonconvex.

MSK_RES_ERR_NONLINEAR_RANGED
: Nonlinear constraints with finite lower and upper bound always define a nonconvex feasible set.

MSK_RES_ERR_CON_Q_NOT_PSD
: The quadratic constraint matrix is not positive semidefinite as expected for a constraint with finite upper bound. This results in a nonconvex problem. The parameter *MSK_DPAR_CHECK_CONVEXITY_REL_TOL* can be used to relax the convexity check.

MSK_RES_ERR_CON_Q_NOT_NSD
: The quadratic constraint matrix is not negative semidefinite as expected for a constraint with finite lower bound. This results in a nonconvex problem. The parameter *MSK_DPAR_CHECK_CONVEXITY_REL_TOL* can be used to relax the convexity check.

MSK_RES_ERR_OBJ_Q_NOT_PSD
: The quadratic coefficient matrix in the objective is not positive semidefinite as expected for a minimization problem. The parameter *MSK_DPAR_CHECK_CONVEXITY_REL_TOL* can be used to relax the convexity check.

MSK_RES_ERR_OBJ_Q_NOT_NSD
: The quadratic coefficient matrix in the objective is not negative semidefinite as expected for a maximization problem. The parameter *MSK_DPAR_CHECK_CONVEXITY_REL_TOL* can be used to relax the convexity check.

MSK_RES_ERR_ARGUMENT_PERM_ARRAY
: An invalid permutation array is specified.

MSK_RES_ERR_CONE_INDEX
An index of a non-existing cone has been specified.

MSK_RES_ERR_CONE_SIZE
A cone with too few members is specified.

MSK_RES_ERR_CONE_OVERLAP
One or more of the variables in the cone to be added is already member of another cone. Now assume the variable is x_j then add a new variable say x_k and the constraint

$$x_j = x_k$$

and then let x_k be member of the cone to be appended.

MSK_RES_ERR_CONE_REP_VAR
A variable is included multiple times in the cone.

MSK_RES_ERR_MAXNUMCONE
The value specified for `maxnumcone` is too small.

MSK_RES_ERR_CONE_TYPE
Invalid cone type specified.

MSK_RES_ERR_CONE_TYPE_STR
Invalid cone type specified.

MSK_RES_ERR_CONE_OVERLAP_APPEND
The cone to be appended has one variable which is already member of another cone.

MSK_RES_ERR_REMOVE_CONE_VARIABLE
A variable cannot be removed because it will make a cone invalid.

MSK_RES_ERR_SOL_FILE_INVALID_NUMBER
An invalid number is specified in a solution file.

MSK_RES_ERR_HUGE_C
A huge value in absolute size is specified for one c_j.

MSK_RES_ERR_HUGE_AIJ
A numerically huge value is specified for an $a_{i,j}$ element in A. The parameter *MSK_DPAR_DATA_TOL_AIJ_HUGE* controls when an $a_{i,j}$ is considered huge.

MSK_RES_ERR_DUPLICATE_AIJ
An element in the A matrix is specified twice.

MSK_RES_ERR_LOWER_BOUND_IS_A_NAN
The lower bound specified is not a number (nan).

MSK_RES_ERR_UPPER_BOUND_IS_A_NAN
The upper bound specified is not a number (nan).

MSK_RES_ERR_INFINITE_BOUND
A numerically huge bound value is specified.

MSK_RES_ERR_INV_QOBJ_SUBI
Invalid value in `qosubi`.

MSK_RES_ERR_INV_QOBJ_SUBJ
Invalid value in `qosubj`.

MSK_RES_ERR_INV_QOBJ_VAL
Invalid value in `qoval`.

MSK_RES_ERR_INV_QCON_SUBK
Invalid value in `qcsubk`.

MSK_RES_ERR_INV_QCON_SUBI
Invalid value in `qcsubi`.

MSK_RES_ERR_INV_QCON_SUBJ
Invalid value in qcsubj.

MSK_RES_ERR_INV_QCON_VAL
Invalid value in qcval.

MSK_RES_ERR_QCON_SUBI_TOO_SMALL
Invalid value in qcsubi.

MSK_RES_ERR_QCON_SUBI_TOO_LARGE
Invalid value in qcsubi.

MSK_RES_ERR_QOBJ_UPPER_TRIANGLE
An element in the upper triangle of Q^o is specified. Only elements in the lower triangle should be specified.

MSK_RES_ERR_QCON_UPPER_TRIANGLE
An element in the upper triangle of a Q^k is specified. Only elements in the lower triangle should be specified.

MSK_RES_ERR_FIXED_BOUND_VALUES
A fixed constraint/variable has been specified using the bound keys but the numerical value of the lower and upper bound is different.

MSK_RES_ERR_NONLINEAR_FUNCTIONS_NOT_ALLOWED
An operation that is invalid for problems with nonlinear functions defined has been attempted.

MSK_RES_ERR_USER_FUNC_RET
An user function reported an error.

MSK_RES_ERR_USER_FUNC_RET_DATA
An user function returned invalid data.

MSK_RES_ERR_USER_NLO_FUNC
The user-defined nonlinear function reported an error.

MSK_RES_ERR_USER_NLO_EVAL
The user-defined nonlinear function reported an error.

MSK_RES_ERR_USER_NLO_EVAL_HESSUBI
The user-defined nonlinear function reported an invalid subscript in the Hessian.

MSK_RES_ERR_USER_NLO_EVAL_HESSUBJ
The user-defined nonlinear function reported an invalid subscript in the Hessian.

MSK_RES_ERR_INVALID_OBJECTIVE_SENSE
An invalid objective sense is specified.

MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE
The objective sense has not been specified before the optimization.

MSK_RES_ERR_Y_IS_UNDEFINED
The solution item y is undefined.

MSK_RES_ERR_NAN_IN_DOUBLE_DATA
An invalid floating point value was used in some double data.

MSK_RES_ERR_NAN_IN_BLC
l^c contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_BUC
u^c contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_C
c contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_BLX
l^x contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_BUX
: u^x contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_INVALID_AIJ
: $a_{i,j}$ contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_SYM_MAT_INVALID
: A symmetric matrix contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_SYM_MAT_HUGE
: A symmetric matrix contains a huge value in absolute size. The parameter *MSK_DPAR_DATA_SYM_MAT_TOL_HUGE* controls when an $e_{i,j}$ is considered huge.

MSK_RES_ERR_INV_PROBLEM
: Invalid problem type. Probably a nonconvex problem has been specified.

MSK_RES_ERR_MIXED_CONIC_AND_NL
: The problem contains nonlinear terms conic constraints. The requested operation cannot be applied to this type of problem.

MSK_RES_ERR_GLOBAL_INV_CONIC_PROBLEM
: The global optimizer can only be applied to problems without semidefinite variables.

MSK_RES_ERR_INV_OPTIMIZER
: An invalid optimizer has been chosen for the problem. This means that the simplex or the conic optimizer is chosen to optimize a nonlinear problem.

MSK_RES_ERR_MIO_NO_OPTIMIZER
: No optimizer is available for the current class of integer optimization problems.

MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE
: No optimizer is available for this class of optimization problems.

MSK_RES_ERR_FINAL_SOLUTION
: An error occurred during the solution finalization.

MSK_RES_ERR_POSTSOLVE
: An error occurred during the postsolve. Please contact **MOSEK** support.

MSK_RES_ERR_OVERFLOW
: A computation produced an overflow i.e. a very large number.

MSK_RES_ERR_NO_BASIS_SOL
: No basic solution is defined.

MSK_RES_ERR_BASIS_FACTOR
: The factorization of the basis is invalid.

MSK_RES_ERR_BASIS_SINGULAR
: The basis is singular and hence cannot be factored.

MSK_RES_ERR_FACTOR
: An error occurred while factorizing a matrix.

MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX
: An optimization problem cannot be relaxed. This is the case e.g. for general nonlinear optimization problems.

MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED
: The relaxed problem could not be solved to optimality. Please consult the log file for further details.

MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND
: The upper bound is less than the lower bound for a variable or a constraint. Please correct this before running the feasibility repair.

MSK_RES_ERR_REPAIR_INVALID_PROBLEM
: The feasibility repair does not support the specified problem type.

MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED
Computation the optimal relaxation failed. The cause may have been numerical problems.

MSK_RES_ERR_NAME_MAX_LEN
A name is longer than the buffer that is supposed to hold it.

MSK_RES_ERR_NAME_IS_NULL
The name buffer is a NULL pointer.

MSK_RES_ERR_INVALID_COMPRESSION
Invalid compression type.

MSK_RES_ERR_INVALID_IOMODE
Invalid io mode.

MSK_RES_ERR_NO_PRIMAL_INFEAS_CER
A certificate of primal infeasibility is not available.

MSK_RES_ERR_NO_DUAL_INFEAS_CER
A certificate of infeasibility is not available.

MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK
The required solution is not available.

MSK_RES_ERR_INV_MARKI
Invalid value in marki.

MSK_RES_ERR_INV_MARKJ
Invalid value in markj.

MSK_RES_ERR_INV_NUMI
Invalid numi.

MSK_RES_ERR_INV_NUMJ
Invalid numj.

MSK_RES_ERR_CANNOT_CLONE_NL
A task with a nonlinear function callback cannot be cloned.

MSK_RES_ERR_CANNOT_HANDLE_NL
A function cannot handle a task with nonlinear function callbacks.

MSK_RES_ERR_INVALID_ACCMODE
An invalid access mode is specified.

MSK_RES_ERR_TASK_INCOMPATIBLE
The Task file is incompatible with this platform. This results from reading a file on a 32 bit platform generated on a 64 bit platform.

MSK_RES_ERR_TASK_INVALID
The Task file is invalid.

MSK_RES_ERR_TASK_WRITE
Failed to write the task file.

MSK_RES_ERR_LU_MAX_NUM_TRIES
Could not compute the LU factors of the matrix within the maximum number of allowed tries.

MSK_RES_ERR_INVALID_UTF8
An invalid UTF8 string is encountered.

MSK_RES_ERR_INVALID_WCHAR
An invalid `wchar` string is encountered.

MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL
No dual information is available for the integer solution.

MSK_RES_ERR_NO_SNX_FOR_BAS_SOL
s_n^x is not available for the basis solution.

`MSK_RES_ERR_INTERNAL`
An internal error occurred. Please report this problem.

`MSK_RES_ERR_API_ARRAY_TOO_SMALL`
An input array was too short.

`MSK_RES_ERR_API_CB_CONNECT`
Failed to connect a callback object.

`MSK_RES_ERR_API_FATAL_ERROR`
An internal error occurred in the API. Please report this problem.

`MSK_RES_ERR_API_INTERNAL`
An internal fatal error occurred in an interface function.

`MSK_RES_ERR_SEN_FORMAT`
Syntax error in sensitivity analysis file.

`MSK_RES_ERR_SEN_UNDEF_NAME`
An undefined name was encountered in the sensitivity analysis file.

`MSK_RES_ERR_SEN_INDEX_RANGE`
Index out of range in the sensitivity analysis file.

`MSK_RES_ERR_SEN_BOUND_INVALID_UP`
Analysis of upper bound requested for an index, where no upper bound exists.

`MSK_RES_ERR_SEN_BOUND_INVALID_LO`
Analysis of lower bound requested for an index, where no lower bound exists.

`MSK_RES_ERR_SEN_INDEX_INVALID`
Invalid range given in the sensitivity file.

`MSK_RES_ERR_SEN_INVALID_REGEXP`
Syntax error in regexp or regexp longer than 1024.

`MSK_RES_ERR_SEN_SOLUTION_STATUS`
No optimal solution found to the original problem given for sensitivity analysis.

`MSK_RES_ERR_SEN_NUMERICAL`
Numerical difficulties encountered performing the sensitivity analysis.

`MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE`
Sensitivity analysis cannot be performed for the specified problem. Sensitivity analysis is only possible for linear problems.

`MSK_RES_ERR_UNB_STEP_SIZE`
A step size in an optimizer was unexpectedly unbounded. For instance, if the step-size becomes unbounded in phase 1 of the simplex algorithm then an error occurs. Normally this will happen only if the problem is badly formulated. Please contact **MOSEK** support if this error occurs.

`MSK_RES_ERR_IDENTICAL_TASKS`
Some tasks related to this function call were identical. Unique tasks were expected.

`MSK_RES_ERR_AD_INVALID_CODELIST`
The code list data was invalid.

`MSK_RES_ERR_INTERNAL_TEST_FAILED`
An internal unit test function failed.

`MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE`
The problem type is not supported by the XML format.

`MSK_RES_ERR_INVALID_AMPL_STUB`
Invalid AMPL stub.

`MSK_RES_ERR_INT64_TO_INT32_CAST`
An 32 bit integer could not cast to a 64 bit integer.

MSK_RES_ERR_SIZE_LICENSE_NUMCORES
The computer contains more cpu cores than the license allows for.

MSK_RES_ERR_INFEAS_UNDEFINED
The requested value is not defined for this solution type.

MSK_RES_ERR_NO_BARX_FOR_SOLUTION
There is no $\overline{X}$ available for the solution specified. In particular note there are no $\overline{X}$ defined for the basic and integer solutions.

MSK_RES_ERR_NO_BARS_FOR_SOLUTION
There is no $\bar{s}$ available for the solution specified. In particular note there are no $\bar{s}$ defined for the basic and integer solutions.

MSK_RES_ERR_BAR_VAR_DIM
The dimension of a symmetric matrix variable has to greater than 0.

MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX
A row index specified for sparse symmetric matrix is invalid.

MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX
A column index specified for sparse symmetric matrix is invalid.

MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR
Only the lower triangular part of sparse symmetric matrix should be specified.

MSK_RES_ERR_SYM_MAT_INVALID_VALUE
The numerical value specified in a sparse symmetric matrix is not a value floating value.

MSK_RES_ERR_SYM_MAT_DUPLICATE
A value in a symmetric matric as been specified more than once.

MSK_RES_ERR_INVALID_SYM_MAT_DIM
A sparse symmetric matrix of invalid dimension is specified.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_SYM_MAT
The file format does not support a problem with symmetric matrix variables.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CONES
The file format does not support a problem with conic constraints.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_GENERAL_NL
The file format does not support a problem with general nonlinear terms.

MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES
Two constraint names are identical.

MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES
Two variable names are identical.

MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES
Two barvariable names are identical.

MSK_RES_ERR_DUPLICATE_CONE_NAMES
Two cone names are identical.

MSK_RES_ERR_NON_UNIQUE_ARRAY
An array does not contain unique elements.

MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE
The value of a function argument is too large.

MSK_RES_ERR_MIO_INTERNAL
A fatal error occurred in the mixed integer optimizer. Please contact **MOSEK** support.

MSK_RES_ERR_INVALID_PROBLEM_TYPE
An invalid problem type.

MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS
Unhandled solution status.

MSK_RES_ERR_UPPER_TRIANGLE
An element in the upper triangle of a lower triangular matrix is specified.

MSK_RES_ERR_LAU_SINGULAR_MATRIX
A matrix is singular.

MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE
A matrix is not positive definite.

MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX
An invalid lower triangular matrix.

MSK_RES_ERR_LAU_UNKNOWN
An unknown error.

MSK_RES_ERR_LAU_ARG_M
Invalid argument m.

MSK_RES_ERR_LAU_ARG_N
Invalid argument n.

MSK_RES_ERR_LAU_ARG_K
Invalid argument k.

MSK_RES_ERR_LAU_ARG_TRANSA
Invalid argument transa.

MSK_RES_ERR_LAU_ARG_TRANSB
Invalid argument transb.

MSK_RES_ERR_LAU_ARG_UPLO
Invalid argument uplo.

MSK_RES_ERR_LAU_ARG_TRANS
Invalid argument trans.

MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX
An invalid sparse symmetric matrix is specfified. Note only the lower triangular part with no duplicates is specifed.

MSK_RES_ERR_CBF_PARSE
An error occurred while parsing an CBF file.

MSK_RES_ERR_CBF_OBJ_SENSE
An invalid objective sense is specified.

MSK_RES_ERR_CBF_NO_VARIABLES
No variables are specified.

MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS
Too many constraints specified.

MSK_RES_ERR_CBF_TOO_MANY_VARIABLES
Too many variables specified.

MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED
No version specified.

MSK_RES_ERR_CBF_SYNTAX
Invalid syntax.

MSK_RES_ERR_CBF_DUPLICATE_OBJ
Duplicate OBJ keyword.

MSK_RES_ERR_CBF_DUPLICATE_CON
Duplicate CON keyword.

MSK_RES_ERR_CBF_DUPLICATE_VAR
Duplicate VAR keyword.

MSK_RES_ERR_CBF_DUPLICATE_INT
Duplicate INT keyword.

MSK_RES_ERR_CBF_INVALID_VAR_TYPE
Invalid variable type.

MSK_RES_ERR_CBF_INVALID_CON_TYPE
Invalid constraint type.

MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION
Invalid domain dimension.

MSK_RES_ERR_CBF_DUPLICATE_OBJACOORD
Duplicate index in OBJCOORD.

MSK_RES_ERR_CBF_DUPLICATE_BCOORD
Duplicate index in BCOORD.

MSK_RES_ERR_CBF_DUPLICATE_ACOORD
Duplicate index in ACOORD.

MSK_RES_ERR_CBF_TOO_FEW_VARIABLES
Too few variables defined.

MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS
Too few constraints defined.

MSK_RES_ERR_CBF_TOO_FEW_INTS
Too few ints are specified.

MSK_RES_ERR_CBF_TOO_MANY_INTS
Too many ints are specified.

MSK_RES_ERR_CBF_INVALID_INT_INDEX
Invalid INT index.

MSK_RES_ERR_CBF_UNSUPPORTED
Unsupported feature is present.

MSK_RES_ERR_CBF_DUPLICATE_PSDVAR
Duplicate PSDVAR keyword.

MSK_RES_ERR_CBF_INVALID_PSDVAR_DIMENSION
Invalid PSDVAR dimmension.

MSK_RES_ERR_CBF_TOO_FEW_PSDVAR
Too few variables defined.

MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER
An invalid root optimizer was selected for the problem type.

MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER
An invalid node optimizer was selected for the problem type.

MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD
The matrix defining the quadratric part of constraint is not positive semidefinite.

MSK_RES_ERR_TOCONIC_CONSTRAINT_FX
The quadratic constraint is an equality, thus not convex.

MSK_RES_ERR_TOCONIC_CONSTRAINT_RA
The quadratic constraint has finite lower and upper bound, and therefore it is not convex.

MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC
The constraint is not conic representable.

MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD
The matrix defining the quadratic part of the objective function is not positive semidefinite.

MSK_RES_ERR_SERVER_CONNECT
Failed to connect to remote solver server. The server string or the port string were invalid, or the server did not accept connection.

MSK_RES_ERR_SERVER_PROTOCOL
Unexpected message or data from solver server.

MSK_RES_ERR_SERVER_STATUS
Server returned non-ok HTTP status code

MSK_RES_ERR_SERVER_TOKEN
The job ID specified is incorrect or invalid

11.5 Constants

11.5.1 Language selection constants

MSK_LANG_ENG
English language selection

MSK_LANG_DAN
Danish language selection

11.5.2 Constraint or variable access modes. All functions using this enum are deprecated. Use separate functions for rows/columns instead.

MSK_ACC_VAR
Access data by columns (variable oriented)

MSK_ACC_CON
Access data by rows (constraint oriented)

11.5.3 Basis identification

MSK_BI_NEVER
Never do basis identification.

MSK_BI_ALWAYS
Basis identification is always performed even if the interior-point optimizer terminates abnormally.

MSK_BI_NO_ERROR
Basis identification is performed if the interior-point optimizer terminates without an error.

MSK_BI_IF_FEASIBLE
Basis identification is not performed if the interior-point optimizer terminates with a problem status saying that the problem is primal or dual infeasible.

MSK_BI_RESERVERED
Not currently in use.

11.5.4 Bound keys

MSK_BK_LO
The constraint or variable has a finite lower bound and an infinite upper bound.

MSK_BK_UP
The constraint or variable has an infinite lower bound and an finite upper bound.

MSK_BK_FX
The constraint or variable is fixed.

MSK_BK_FR
The constraint or variable is free.

MSK_BK_RA
The constraint or variable is ranged.

11.5.5 Mark

MSK_MARK_LO
The lower bound is selected for sensitivity analysis.

MSK_MARK_UP
The upper bound is selected for sensitivity analysis.

11.5.6 Degeneracy strategies

MSK_SIM_DEGEN_NONE
The simplex optimizer should use no degeneration strategy.

MSK_SIM_DEGEN_FREE
The simplex optimizer chooses the degeneration strategy.

MSK_SIM_DEGEN_AGGRESSIVE
The simplex optimizer should use an aggressive degeneration strategy.

MSK_SIM_DEGEN_MODERATE
The simplex optimizer should use a moderate degeneration strategy.

MSK_SIM_DEGEN_MINIMUM
The simplex optimizer should use a minimum degeneration strategy.

11.5.7 Transposed matrix.

MSK_TRANSPOSE_NO
No transpose is applied.

MSK_TRANSPOSE_YES
A transpose is applied.

11.5.8 Triangular part of a symmetric matrix.

MSK_UPLO_LO
Lower part.

MSK_UPLO_UP
Upper part

11.5.9 Problem reformulation.

MSK_SIM_REFORMULATION_ON
Allow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_OFF
Disallow the simplex optimizer to reformulate the problem.

MSK_SIM_REFORMULATION_FREE
The simplex optimizer can choose freely.

MSK_SIM_REFORMULATION_AGGRESSIVE
The simplex optimizer should use an aggressive reformulation strategy.

11.5.10 Exploit duplicate columns.

MSK_SIM_EXPLOIT_DUPVEC_ON
Allow the simplex optimizer to exploit duplicated columns.

MSK_SIM_EXPLOIT_DUPVEC_OFF
Disallow the simplex optimizer to exploit duplicated columns.

MSK_SIM_EXPLOIT_DUPVEC_FREE
The simplex optimizer can choose freely.

11.5.11 Hot-start type employed by the simplex optimizer

MSK_SIM_HOTSTART_NONE
The simplex optimizer performs a coldstart.

MSK_SIM_HOTSTART_FREE
The simplex optimize chooses the hot-start type.

MSK_SIM_HOTSTART_STATUS_KEYS
Only the status keys of the constraints and variables are used to choose the type of hot-start.

11.5.12 Hot-start type employed by the interior-point optimizers.

MSK_INTPNT_HOTSTART_NONE
The interior-point optimizer performs a coldstart.

MSK_INTPNT_HOTSTART_PRIMAL
The interior-point optimizer exploits the primal solution only.

MSK_INTPNT_HOTSTART_DUAL
The interior-point optimizer exploits the dual solution only.

MSK_INTPNT_HOTSTART_PRIMAL_DUAL
The interior-point optimizer exploits both the primal and dual solution.

11.5.13 Progress callback codes

MSK_CALLBACK_BEGIN_BI
The basis identification procedure has been started.

MSK_CALLBACK_BEGIN_CONIC
The callback function is called when the conic optimizer is started.

MSK_CALLBACK_BEGIN_DUAL_BI
The callback function is called from within the basis identification procedure when the dual phase is started.

MSK_CALLBACK_BEGIN_DUAL_SENSITIVITY
Dual sensitivity analysis is started.

MSK_CALLBACK_BEGIN_DUAL_SETUP_BI
The callback function is called when the dual BI phase is started.

MSK_CALLBACK_BEGIN_DUAL_SIMPLEX
The callback function is called when the dual simplex optimizer started.

MSK_CALLBACK_BEGIN_DUAL_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the dual simplex clean-up phase is started.

MSK_CALLBACK_BEGIN_FULL_CONVEXITY_CHECK
Begin full convexity check.

MSK_CALLBACK_BEGIN_INFEAS_ANA
The callback function is called when the infeasibility analyzer is started.

MSK_CALLBACK_BEGIN_INTPNT
The callback function is called when the interior-point optimizer is started.

MSK_CALLBACK_BEGIN_LICENSE_WAIT
Begin waiting for license.

MSK_CALLBACK_BEGIN_MIO
The callback function is called when the mixed-integer optimizer is started.

MSK_CALLBACK_BEGIN_OPTIMIZER
The callback function is called when the optimizer is started.

MSK_CALLBACK_BEGIN_PRESOLVE
The callback function is called when the presolve is started.

MSK_CALLBACK_BEGIN_PRIMAL_BI
The callback function is called from within the basis identification procedure when the primal phase is started.

MSK_CALLBACK_BEGIN_PRIMAL_REPAIR
Begin primal feasibility repair.

MSK_CALLBACK_BEGIN_PRIMAL_SENSITIVITY
Primal sensitivity analysis is started.

MSK_CALLBACK_BEGIN_PRIMAL_SETUP_BI
The callback function is called when the primal BI setup is started.

MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX
The callback function is called when the primal simplex optimizer is started.

MSK_CALLBACK_BEGIN_PRIMAL_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the primal simplex clean-up phase is started.

MSK_CALLBACK_BEGIN_QCQO_REFORMULATE
Begin QCQO reformulation.

MSK_CALLBACK_BEGIN_READ
MOSEK has started reading a problem file.

MSK_CALLBACK_BEGIN_ROOT_CUTGEN
The callback function is called when root cut generation is started.

MSK_CALLBACK_BEGIN_SIMPLEX
The callback function is called when the simplex optimizer is started.

MSK_CALLBACK_BEGIN_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the simplex clean-up phase is started.

MSK_CALLBACK_BEGIN_TO_CONIC
Begin conic reformulation.

MSK_CALLBACK_BEGIN_WRITE
: **MOSEK** has started writing a problem file.

MSK_CALLBACK_CONIC
: The callback function is called from within the conic optimizer after the information database has been updated.

MSK_CALLBACK_DUAL_SIMPLEX
: The callback function is called from within the dual simplex optimizer.

MSK_CALLBACK_END_BI
: The callback function is called when the basis identification procedure is terminated.

MSK_CALLBACK_END_CONIC
: The callback function is called when the conic optimizer is terminated.

MSK_CALLBACK_END_DUAL_BI
: The callback function is called from within the basis identification procedure when the dual phase is terminated.

MSK_CALLBACK_END_DUAL_SENSITIVITY
: Dual sensitivity analysis is terminated.

MSK_CALLBACK_END_DUAL_SETUP_BI
: The callback function is called when the dual BI phase is terminated.

MSK_CALLBACK_END_DUAL_SIMPLEX
: The callback function is called when the dual simplex optimizer is terminated.

MSK_CALLBACK_END_DUAL_SIMPLEX_BI
: The callback function is called from within the basis identification procedure when the dual clean-up phase is terminated.

MSK_CALLBACK_END_FULL_CONVEXITY_CHECK
: End full convexity check.

MSK_CALLBACK_END_INFEAS_ANA
: The callback function is called when the infeasibility analyzer is terminated.

MSK_CALLBACK_END_INTPNT
: The callback function is called when the interior-point optimizer is terminated.

MSK_CALLBACK_END_LICENSE_WAIT
: End waiting for license.

MSK_CALLBACK_END_MIO
: The callback function is called when the mixed-integer optimizer is terminated.

MSK_CALLBACK_END_OPTIMIZER
: The callback function is called when the optimizer is terminated.

MSK_CALLBACK_END_PRESOLVE
: The callback function is called when the presolve is completed.

MSK_CALLBACK_END_PRIMAL_BI
: The callback function is called from within the basis identification procedure when the primal phase is terminated.

MSK_CALLBACK_END_PRIMAL_REPAIR
: End primal feasibility repair.

MSK_CALLBACK_END_PRIMAL_SENSITIVITY
: Primal sensitivity analysis is terminated.

MSK_CALLBACK_END_PRIMAL_SETUP_BI
: The callback function is called when the primal BI setup is terminated.

MSK_CALLBACK_END_PRIMAL_SIMPLEX
: The callback function is called when the primal simplex optimizer is terminated.

MSK_CALLBACK_END_PRIMAL_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the primal clean-up phase is terminated.

MSK_CALLBACK_END_QCQO_REFORMULATE
End QCQO reformulation.

MSK_CALLBACK_END_READ
MOSEK has finished reading a problem file.

MSK_CALLBACK_END_ROOT_CUTGEN
The callback function is called when root cut generation is is terminated.

MSK_CALLBACK_END_SIMPLEX
The callback function is called when the simplex optimizer is terminated.

MSK_CALLBACK_END_SIMPLEX_BI
The callback function is called from within the basis identification procedure when the simplex clean-up phase is terminated.

MSK_CALLBACK_END_TO_CONIC
End conic reformulation.

MSK_CALLBACK_END_WRITE
MOSEK has finished writing a problem file.

MSK_CALLBACK_IM_BI
The callback function is called from within the basis identification procedure at an intermediate point.

MSK_CALLBACK_IM_CONIC
The callback function is called at an intermediate stage within the conic optimizer where the information database has not been updated.

MSK_CALLBACK_IM_DUAL_BI
The callback function is called from within the basis identification procedure at an intermediate point in the dual phase.

MSK_CALLBACK_IM_DUAL_SENSIVITY
The callback function is called at an intermediate stage of the dual sensitivity analysis.

MSK_CALLBACK_IM_DUAL_SIMPLEX
The callback function is called at an intermediate point in the dual simplex optimizer.

MSK_CALLBACK_IM_FULL_CONVEXITY_CHECK
The callback function is called at an intermediate stage of the full convexity check.

MSK_CALLBACK_IM_INTPNT
The callback function is called at an intermediate stage within the interior-point optimizer where the information database has not been updated.

MSK_CALLBACK_IM_LICENSE_WAIT
MOSEK is waiting for a license.

MSK_CALLBACK_IM_LU
The callback function is called from within the LU factorization procedure at an intermediate point.

MSK_CALLBACK_IM_MIO
The callback function is called at an intermediate point in the mixed-integer optimizer.

MSK_CALLBACK_IM_MIO_DUAL_SIMPLEX
The callback function is called at an intermediate point in the mixed-integer optimizer while running the dual simplex optimizer.

MSK_CALLBACK_IM_MIO_INTPNT
The callback function is called at an intermediate point in the mixed-integer optimizer while running the interior-point optimizer.

MSK_CALLBACK_IM_MIO_PRIMAL_SIMPLEX
: The callback function is called at an intermediate point in the mixed-integer optimizer while running the primal simplex optimizer.

MSK_CALLBACK_IM_ORDER
: The callback function is called from within the matrix ordering procedure at an intermediate point.

MSK_CALLBACK_IM_PRESOLVE
: The callback function is called from within the presolve procedure at an intermediate stage.

MSK_CALLBACK_IM_PRIMAL_BI
: The callback function is called from within the basis identification procedure at an intermediate point in the primal phase.

MSK_CALLBACK_IM_PRIMAL_SENSIVITY
: The callback function is called at an intermediate stage of the primal sensitivity analysis.

MSK_CALLBACK_IM_PRIMAL_SIMPLEX
: The callback function is called at an intermediate point in the primal simplex optimizer.

MSK_CALLBACK_IM_QO_REFORMULATE
: The callback function is called at an intermediate stage of the conic quadratic reformulation.

MSK_CALLBACK_IM_READ
: Intermediate stage in reading.

MSK_CALLBACK_IM_ROOT_CUTGEN
: The callback is called from within root cut generation at an intermediate stage.

MSK_CALLBACK_IM_SIMPLEX
: The callback function is called from within the simplex optimizer at an intermediate point.

MSK_CALLBACK_IM_SIMPLEX_BI
: The callback function is called from within the basis identification procedure at an intermediate point in the simplex clean-up phase. The frequency of the callbacks is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

MSK_CALLBACK_INTPNT
: The callback function is called from within the interior-point optimizer after the information database has been updated.

MSK_CALLBACK_NEW_INT_MIO
: The callback function is called after a new integer solution has been located by the mixed-integer optimizer.

MSK_CALLBACK_PRIMAL_SIMPLEX
: The callback function is called from within the primal simplex optimizer.

MSK_CALLBACK_READ_OPF
: The callback function is called from the OPF reader.

MSK_CALLBACK_READ_OPF_SECTION
: A chunk of Q non-zeros has been read from a problem file.

MSK_CALLBACK_SOLVING_REMOTE
: The callback function is called while the task is being solved on a remote server.

MSK_CALLBACK_UPDATE_DUAL_BI
: The callback function is called from within the basis identification procedure at an intermediate point in the dual phase.

MSK_CALLBACK_UPDATE_DUAL_SIMPLEX
: The callback function is called in the dual simplex optimizer.

MSK_CALLBACK_UPDATE_DUAL_SIMPLEX_BI
: The callback function is called from within the basis identification procedure at an intermediate point in the dual simplex clean-up phase. The frequency of the callbacks is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

MSK_CALLBACK_UPDATE_PRESOLVE
The callback function is called from within the presolve procedure.

MSK_CALLBACK_UPDATE_PRIMAL_BI
The callback function is called from within the basis identification procedure at an intermediate point in the primal phase.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX
The callback function is called in the primal simplex optimizer.

MSK_CALLBACK_UPDATE_PRIMAL_SIMPLEX_BI
The callback function is called from within the basis identification procedure at an intermediate point in the primal simplex clean-up phase. The frequency of the callbacks is controlled by the *MSK_IPAR_LOG_SIM_FREQ* parameter.

MSK_CALLBACK_WRITE_OPF
The callback function is called from the OPF writer.

11.5.14 Types of convexity checks.

MSK_CHECK_CONVEXITY_NONE
No convexity check.

MSK_CHECK_CONVEXITY_SIMPLE
Perform simple and fast convexity check.

MSK_CHECK_CONVEXITY_FULL
Perform a full convexity check.

11.5.15 Compression types

MSK_COMPRESS_NONE
No compression is used.

MSK_COMPRESS_FREE
The type of compression used is chosen automatically.

MSK_COMPRESS_GZIP
The type of compression used is gzip compatible.

11.5.16 Cone types

MSK_CT_QUAD
The cone is a quadratic cone.

MSK_CT_RQUAD
The cone is a rotated quadratic cone.

11.5.17 Name types

MSK_NAME_TYPE_GEN
General names. However, no duplicate and blank names are allowed.

MSK_NAME_TYPE_MPS
MPS type names.

MSK_NAME_TYPE_LP
LP type names.

11.5.18 Cone types

MSK_SYMMAT_TYPE_SPARSE
Sparse symmetric matrix.

11.5.19 Data format types

MSK_DATA_FORMAT_EXTENSION
The file extension is used to determine the data file format.

MSK_DATA_FORMAT_MPS
The data file is MPS formatted.

MSK_DATA_FORMAT_LP
The data file is LP formatted.

MSK_DATA_FORMAT_OP
The data file is an optimization problem formatted file.

MSK_DATA_FORMAT_XML
The data file is an XML formatted file.

MSK_DATA_FORMAT_FREE_MPS
The data a free MPS formatted file.

MSK_DATA_FORMAT_TASK
Generic task dump file.

MSK_DATA_FORMAT_CB
Conic benchmark format,

MSK_DATA_FORMAT_JSON_TASK
JSON based task format.

11.5.20 Double information items

MSK_DINF_BI_CLEAN_DUAL_TIME
Time spent within the dual clean-up optimizer of the basis identification procedure since its invocation.

MSK_DINF_BI_CLEAN_PRIMAL_TIME
Time spent within the primal clean-up optimizer of the basis identification procedure since its invocation.

MSK_DINF_BI_CLEAN_TIME
Time spent within the clean-up phase of the basis identification procedure since its invocation.

MSK_DINF_BI_DUAL_TIME
Time spent within the dual phase basis identification procedure since its invocation.

MSK_DINF_BI_PRIMAL_TIME
Time spent within the primal phase of the basis identification procedure since its invocation.

MSK_DINF_BI_TIME
Time spent within the basis identification procedure since its invocation.

MSK_DINF_INTPNT_DUAL_FEAS
Dual feasibility measure reported by the interior-point optimizer. (For the interior-point optimizer this measure is not directly related to the original problem because a homogeneous model is employed.)

MSK_DINF_INTPNT_DUAL_OBJ
Dual objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_FACTOR_NUM_FLOPS
An estimate of the number of flops used in the factorization.

MSK_DINF_INTPNT_OPT_STATUS
A measure of optimality of the solution. It should converge to $+1$ if the problem has a primal-dual optimal solution, and converge to -1 if the problem is (strictly) primal or dual infeasible. If the measure converges to another constant, or fails to settle, the problem is usually ill-posed.

MSK_DINF_INTPNT_ORDER_TIME
Order time (in seconds).

MSK_DINF_INTPNT_PRIMAL_FEAS
Primal feasibility measure reported by the interior-point optimizer. (For the interior-point optimizer this measure is not directly related to the original problem because a homogeneous model is employed).

MSK_DINF_INTPNT_PRIMAL_OBJ
Primal objective value reported by the interior-point optimizer.

MSK_DINF_INTPNT_TIME
Time spent within the interior-point optimizer since its invocation.

MSK_DINF_MIO_CLIQUE_SEPARATION_TIME
Seperation time for clique cuts.

MSK_DINF_MIO_CMIR_SEPARATION_TIME
Seperation time for CMIR cuts.

MSK_DINF_MIO_CONSTRUCT_SOLUTION_OBJ
If **MOSEK** has successfully constructed an integer feasible solution, then this item contains the optimal objective value corresponding to the feasible solution.

MSK_DINF_MIO_DUAL_BOUND_AFTER_PRESOLVE
Value of the dual bound after presolve but before cut generation.

MSK_DINF_MIO_GMI_SEPARATION_TIME
Seperation time for GMI cuts.

MSK_DINF_MIO_HEURISTIC_TIME
Total time spent in the optimizer.

MSK_DINF_MIO_IMPLIED_BOUND_TIME
Seperation time for implied bound cuts.

MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME
Seperation time for knapsack cover.

MSK_DINF_MIO_OBJ_ABS_GAP
Given the mixed-integer optimizer has computed a feasible solution and a bound on the optimal objective value, then this item contains the absolute gap defined by

$$|(\text{objective value of feasible solution}) - (\text{objective bound})|.$$

Otherwise it has the value -1.0.

MSK_DINF_MIO_OBJ_BOUND
The best known bound on the objective function. This value is undefined until at least one relaxation has been solved: To see if this is the case check that *MSK_IINF_MIO_NUM_RELAX* is strictly positive.

MSK_DINF_MIO_OBJ_INT
The primal objective value corresponding to the best integer feasible solution. Please note that at least one integer feasible solution must have been located i.e. check *MSK_IINF_MIO_NUM_INT_SOLUTIONS*.

`MSK_DINF_MIO_OBJ_REL_GAP`
Given that the mixed-integer optimizer has computed a feasible solution and a bound on the optimal objective value, then this item contains the relative gap defined by

$$\frac{|(\text{objective value of feasible solution}) - (\text{objective bound})|}{\max(\delta, |(\text{objective value of feasible solution})|)}.$$

where δ is given by the parameter *MSK_DPAR_MIO_REL_GAP_CONST*. Otherwise it has the value -1.0.

`MSK_DINF_MIO_OPTIMIZER_TIME`
Total time spent in the optimizer.

`MSK_DINF_MIO_PROBING_TIME`
Total time for probing.

`MSK_DINF_MIO_ROOT_CUTGEN_TIME`
Total time for cut generation.

`MSK_DINF_MIO_ROOT_OPTIMIZER_TIME`
Time spent in the optimizer while solving the root relaxation.

`MSK_DINF_MIO_ROOT_PRESOLVE_TIME`
Time spent in while presolving the root relaxation.

`MSK_DINF_MIO_TIME`
Time spent in the mixed-integer optimizer.

`MSK_DINF_MIO_USER_OBJ_CUT`
If the objective cut is used, then this information item has the value of the cut.

`MSK_DINF_OPTIMIZER_TIME`
Total time spent in the optimizer since it was invoked.

`MSK_DINF_PRESOLVE_ELI_TIME`
Total time spent in the eliminator since the presolve was invoked.

`MSK_DINF_PRESOLVE_LINDEP_TIME`
Total time spent in the linear dependency checker since the presolve was invoked.

`MSK_DINF_PRESOLVE_TIME`
Total time (in seconds) spent in the presolve since it was invoked.

`MSK_DINF_PRIMAL_REPAIR_PENALTY_OBJ`
The optimal objective value of the penalty function.

`MSK_DINF_QCQO_REFORMULATE_MAX_PERTURBATION`
Maximum absolute diagonal perturbation occuring during the QCQO reformulation.

`MSK_DINF_QCQO_REFORMULATE_TIME`
Time spent with conic quadratic reformulation.

`MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_COLUMN_SCALING`
Worst Cholesky column scaling.

`MSK_DINF_QCQO_REFORMULATE_WORST_CHOLESKY_DIAG_SCALING`
Worst Cholesky diagonal scaling.

`MSK_DINF_RD_TIME`
Time spent reading the data file.

`MSK_DINF_SIM_DUAL_TIME`
Time spent in the dual simplex optimizer since invoking it.

`MSK_DINF_SIM_FEAS`
Feasibility measure reported by the simplex optimizer.

`MSK_DINF_SIM_OBJ`
Objective value reported by the simplex optimizer.

MSK_DINF_SIM_PRIMAL_TIME
Time spent in the primal simplex optimizer since invoking it.

MSK_DINF_SIM_TIME
Time spent in the simplex optimizer since invoking it.

MSK_DINF_SOL_BAS_DUAL_OBJ
Dual objective value of the basic solution.

MSK_DINF_SOL_BAS_DVIOLCON
Maximal dual bound violation for x^c in the basic solution.

MSK_DINF_SOL_BAS_DVIOLVAR
Maximal dual bound violation for x^x in the basic solution.

MSK_DINF_SOL_BAS_NRM_BARX
Infinity norm of $\overline{X}$ in the basic solution.

MSK_DINF_SOL_BAS_NRM_SLC
Infinity norm of s_l^c in the basic solution.

MSK_DINF_SOL_BAS_NRM_SLX
Infinity norm of s_l^x in the basic solution.

MSK_DINF_SOL_BAS_NRM_SUC
Infinity norm of s_u^c in the basic solution.

MSK_DINF_SOL_BAS_NRM_SUX
Infinity norm of s_u^X in the basic solution.

MSK_DINF_SOL_BAS_NRM_XC
Infinity norm of x^c in the basic solution.

MSK_DINF_SOL_BAS_NRM_XX
Infinity norm of x^x in the basic solution.

MSK_DINF_SOL_BAS_NRM_Y
Infinity norm of y in the basic solution.

MSK_DINF_SOL_BAS_PRIMAL_OBJ
Primal objective value of the basic solution.

MSK_DINF_SOL_BAS_PVIOLCON
Maximal primal bound violation for x^c in the basic solution.

MSK_DINF_SOL_BAS_PVIOLVAR
Maximal primal bound violation for x^x in the basic solution.

MSK_DINF_SOL_ITG_NRM_BARX
Infinity norm of $\overline{X}$ in the integer solution.

MSK_DINF_SOL_ITG_NRM_XC
Infinity norm of x^c in the integer solution.

MSK_DINF_SOL_ITG_NRM_XX
Infinity norm of x^x in the integer solution.

MSK_DINF_SOL_ITG_PRIMAL_OBJ
Primal objective value of the integer solution.

MSK_DINF_SOL_ITG_PVIOLBARVAR
Maximal primal bound violation for $\overline{X}$ in the integer solution.

MSK_DINF_SOL_ITG_PVIOLCON
Maximal primal bound violation for x^c in the integer solution.

MSK_DINF_SOL_ITG_PVIOLCONES
Maximal primal violation for primal conic constraints in the integer solution.

MSK_DINF_SOL_ITG_PVIOLITG
: Maximal violation for the integer constraints in the integer solution.

MSK_DINF_SOL_ITG_PVIOLVAR
: Maximal primal bound violation for x^x in the integer solution.

MSK_DINF_SOL_ITR_DUAL_OBJ
: Dual objective value of the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLBARVAR
: Maximal dual bound violation for $\overline{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLCON
: Maximal dual bound violation for x^c in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLCONES
: Maximal dual violation for dual conic constraints in the interior-point solution.

MSK_DINF_SOL_ITR_DVIOLVAR
: Maximal dual bound violation for x^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_BARS
: Infinity norm of $\overline{S}$ in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_BARX
: Infinity norm of $\overline{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SLC
: Infinity norm of s_l^c in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SLX
: Infinity norm of s_l^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SNX
: Infinity norm of s_n^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SUC
: Infinity norm of s_u^c in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_SUX
: Infinity norm of s_u^X in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_XC
: Infinity norm of x^c in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_XX
: Infinity norm of x^x in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_Y
: Infinity norm of y in the interior-point solution.

MSK_DINF_SOL_ITR_PRIMAL_OBJ
: Primal objective value of the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLBARVAR
: Maximal primal bound violation for $\overline{X}$ in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLCON
: Maximal primal bound violation for x^c in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLCONES
: Maximal primal violation for primal conic constraints in the interior-point solution.

MSK_DINF_SOL_ITR_PVIOLVAR
: Maximal primal bound violation for x^x in the interior-point solution.

MSK_DINF_TO_CONIC_TIME
: Time spent in the last to conic reformulation.

11.5.21 License feature

MSK_FEATURE_PTS
Base system.

MSK_FEATURE_PTON
Nonlinear extension.

11.5.22 Long integer information items.

MSK_LIINF_BI_CLEAN_DUAL_DEG_ITER
Number of dual degenerate clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_DUAL_ITER
Number of dual clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_DEG_ITER
Number of primal degenerate clean iterations performed in the basis identification.

MSK_LIINF_BI_CLEAN_PRIMAL_ITER
Number of primal clean iterations performed in the basis identification.

MSK_LIINF_BI_DUAL_ITER
Number of dual pivots performed in the basis identification.

MSK_LIINF_BI_PRIMAL_ITER
Number of primal pivots performed in the basis identification.

MSK_LIINF_INTPNT_FACTOR_NUM_NZ
Number of non-zeros in factorization.

MSK_LIINF_MIO_INTPNT_ITER
Number of interior-point iterations performed by the mixed-integer optimizer.

MSK_LIINF_MIO_PRESOLVED_ANZ
Number of non-zero entries in the constraint matrix of presolved problem.

MSK_LIINF_MIO_SIM_MAXITER_SETBACKS
Number of times the the simplex optimizer has hit the maximum iteration limit when re-optimizing.

MSK_LIINF_MIO_SIMPLEX_ITER
Number of simplex iterations performed by the mixed-integer optimizer.

MSK_LIINF_RD_NUMANZ
Number of non-zeros in A that is read.

MSK_LIINF_RD_NUMQNZ
Number of Q non-zeros.

11.5.23 Integer information items.

MSK_IINF_ANA_PRO_NUM_CON
Number of constraints in the problem.

MSK_IINF_ANA_PRO_NUM_CON_EQ
Number of equality constraints.

MSK_IINF_ANA_PRO_NUM_CON_FR
Number of unbounded constraints.

MSK_IINF_ANA_PRO_NUM_CON_LO
Number of constraints with a lower bound and an infinite upper bound.

MSK_IINF_ANA_PRO_NUM_CON_RA
Number of constraints with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_CON_UP
Number of constraints with an upper bound and an infinite lower bound.

MSK_IINF_ANA_PRO_NUM_VAR
Number of variables in the problem.

MSK_IINF_ANA_PRO_NUM_VAR_BIN
Number of binary (0-1) variables.

MSK_IINF_ANA_PRO_NUM_VAR_CONT
Number of continuous variables.

MSK_IINF_ANA_PRO_NUM_VAR_EQ
Number of fixed variables.

MSK_IINF_ANA_PRO_NUM_VAR_FR
Number of free variables.

MSK_IINF_ANA_PRO_NUM_VAR_INT
Number of general integer variables.

MSK_IINF_ANA_PRO_NUM_VAR_LO
Number of variables with a lower bound and an infinite upper bound.

MSK_IINF_ANA_PRO_NUM_VAR_RA
Number of variables with finite lower and upper bounds.

MSK_IINF_ANA_PRO_NUM_VAR_UP
Number of variables with an upper bound and an infinite lower bound. This value is set by

MSK_IINF_INTPNT_FACTOR_DIM_DENSE
Dimension of the dense sub system in factorization.

MSK_IINF_INTPNT_ITER
Number of interior-point iterations since invoking the interior-point optimizer.

MSK_IINF_INTPNT_NUM_THREADS
Number of threads that the interior-point optimizer is using.

MSK_IINF_INTPNT_SOLVE_DUAL
Non-zero if the interior-point optimizer is solving the dual problem.

MSK_IINF_MIO_ABSGAP_SATISFIED
Non-zero if absolute gap is within tolerances.

MSK_IINF_MIO_CLIQUE_TABLE_SIZE
Size of the clique table.

MSK_IINF_MIO_CONSTRUCT_NUM_ROUNDINGS
Number of values in the integer solution that is rounded to an integer value.

MSK_IINF_MIO_CONSTRUCT_SOLUTION
If this item has the value 0, then **MOSEK** did not try to construct an initial integer feasible solution. If the item has a positive value, then **MOSEK** successfully constructed an initial integer feasible solution.

MSK_IINF_MIO_INITIAL_SOLUTION
Is non-zero if an initial integer solution is specified.

MSK_IINF_MIO_NEAR_ABSGAP_SATISFIED
Non-zero if absolute gap is within relaxed tolerances.

MSK_IINF_MIO_NEAR_RELGAP_SATISFIED
Non-zero if relative gap is within relaxed tolerances.

MSK_IINF_MIO_NODE_DEPTH
Depth of the last node solved.

MSK_IINF_MIO_NUM_ACTIVE_NODES
: Number of active branch bound nodes.

MSK_IINF_MIO_NUM_BRANCH
: Number of branches performed during the optimization.

MSK_IINF_MIO_NUM_CLIQUE_CUTS
: Number of clique cuts.

MSK_IINF_MIO_NUM_CMIR_CUTS
: Number of Complemented Mixed Integer Rounding (CMIR) cuts.

MSK_IINF_MIO_NUM_GOMORY_CUTS
: Number of Gomory cuts.

MSK_IINF_MIO_NUM_IMPLIED_BOUND_CUTS
: Number of implied bound cuts.

MSK_IINF_MIO_NUM_INT_SOLUTIONS
: Number of integer feasible solutions that has been found.

MSK_IINF_MIO_NUM_KNAPSACK_COVER_CUTS
: Number of clique cuts.

MSK_IINF_MIO_NUM_RELAX
: Number of relaxations solved during the optimization.

MSK_IINF_MIO_NUM_REPEATED_PRESOLVE
: Number of times presolve was repeated at root.

MSK_IINF_MIO_NUMCON
: Number of constraints in the problem solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMINT
: Number of integer variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_NUMVAR
: Number of variables in the problem solved by the mixed-integer optimizer.

MSK_IINF_MIO_OBJ_BOUND_DEFINED
: Non-zero if a valid objective bound has been found, otherwise zero.

MSK_IINF_MIO_PRESOLVED_NUMBIN
: Number of binary variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_PRESOLVED_NUMCON
: Number of constraints in the presolved problem.

MSK_IINF_MIO_PRESOLVED_NUMCONT
: Number of continuous variables in the problem solved be the mixed-integer optimizer.

MSK_IINF_MIO_PRESOLVED_NUMINT
: Number of integer variables in the presolved problem.

MSK_IINF_MIO_PRESOLVED_NUMVAR
: Number of variables in the presolved problem.

MSK_IINF_MIO_RELGAP_SATISFIED
: Non-zero if relative gap is within tolerances.

MSK_IINF_MIO_TOTAL_NUM_CUTS
: Total number of cuts generated by the mixed-integer optimizer.

MSK_IINF_MIO_USER_OBJ_CUT
: If it is non-zero, then the objective cut is used.

MSK_IINF_OPT_NUMCON
: Number of constraints in the problem solved when the optimizer is called.

MSK_IINF_OPT_NUMVAR
: Number of variables in the problem solved when the optimizer is called

MSK_IINF_OPTIMIZE_RESPONSE
: The response code returned by optimize.

MSK_IINF_RD_NUMBARVAR
: Number of variables read.

MSK_IINF_RD_NUMCON
: Number of constraints read.

MSK_IINF_RD_NUMCONE
: Number of conic constraints read.

MSK_IINF_RD_NUMINTVAR
: Number of integer-constrained variables read.

MSK_IINF_RD_NUMQ
: Number of nonempty Q matrices read.

MSK_IINF_RD_NUMVAR
: Number of variables read.

MSK_IINF_RD_PROTYPE
: Problem type.

MSK_IINF_SIM_DUAL_DEG_ITER
: The number of dual degenerate iterations.

MSK_IINF_SIM_DUAL_HOTSTART
: If 1 then the dual simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_DUAL_HOTSTART_LU
: If 1 then a valid basis factorization of full rank was located and used by the dual simplex algorithm.

MSK_IINF_SIM_DUAL_INF_ITER
: The number of iterations taken with dual infeasibility.

MSK_IINF_SIM_DUAL_ITER
: Number of dual simplex iterations during the last optimization.

MSK_IINF_SIM_NUMCON
: Number of constraints in the problem solved by the simplex optimizer.

MSK_IINF_SIM_NUMVAR
: Number of variables in the problem solved by the simplex optimizer.

MSK_IINF_SIM_PRIMAL_DEG_ITER
: The number of primal degenerate iterations.

MSK_IINF_SIM_PRIMAL_HOTSTART
: If 1 then the primal simplex algorithm is solving from an advanced basis.

MSK_IINF_SIM_PRIMAL_HOTSTART_LU
: If 1 then a valid basis factorization of full rank was located and used by the primal simplex algorithm.

MSK_IINF_SIM_PRIMAL_INF_ITER
: The number of iterations taken with primal infeasibility.

MSK_IINF_SIM_PRIMAL_ITER
: Number of primal simplex iterations during the last optimization.

MSK_IINF_SIM_SOLVE_DUAL
: Is non-zero if dual problem is solved.

MSK_IINF_SOL_BAS_PROSTA
: Problem status of the basic solution. Updated after each optimization.

MSK_IINF_SOL_BAS_SOLSTA
Solution status of the basic solution. Updated after each optimization.

MSK_IINF_SOL_ITG_PROSTA
Problem status of the integer solution. Updated after each optimization.

MSK_IINF_SOL_ITG_SOLSTA
Solution status of the integer solution. Updated after each optimization.

MSK_IINF_SOL_ITR_PROSTA
Problem status of the interior-point solution. Updated after each optimization.

MSK_IINF_SOL_ITR_SOLSTA
Solution status of the interior-point solution. Updated after each optimization.

MSK_IINF_STO_NUM_A_REALLOC
Number of times the storage for storing A has been changed. A large value may indicates that memory fragmentation may occur.

11.5.24 Information item types

MSK_INF_DOU_TYPE
Is a double information type.

MSK_INF_INT_TYPE
Is an integer.

MSK_INF_LINT_TYPE
Is a long integer.

11.5.25 Input/output modes

MSK_IOMODE_READ
The file is read-only.

MSK_IOMODE_WRITE
The file is write-only. If the file exists then it is truncated when it is opened. Otherwise it is created when it is opened.

MSK_IOMODE_READWRITE
The file is to read and written.

11.5.26 Specifies the branching direction.

MSK_BRANCH_DIR_FREE
The mixed-integer optimizer decides which branch to choose.

MSK_BRANCH_DIR_UP
The mixed-integer optimizer always chooses the up branch first.

MSK_BRANCH_DIR_DOWN
The mixed-integer optimizer always chooses the down branch first.

MSK_BRANCH_DIR_NEAR
Branch in direction nearest to selected fractional variable.

MSK_BRANCH_DIR_FAR
Branch in direction farthest from selected fractional variable.

MSK_BRANCH_DIR_ROOT_LP
Chose direction based on root lp value of selected variable.

MSK_BRANCH_DIR_GUIDED
Branch in direction of current incumbent.

MSK_BRANCH_DIR_PSEUDOCOST
Branch based on the pseudocost of the variable.

11.5.27 Continuous mixed-integer solution type

MSK_MIO_CONT_SOL_NONE
No interior-point or basic solution are reported when the mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ROOT
The reported interior-point and basic solutions are a solution to the root node problem when mixed-integer optimizer is used.

MSK_MIO_CONT_SOL_ITG
The reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. A solution is only reported in case the problem has a primal feasible solution.

MSK_MIO_CONT_SOL_ITG_REL
In case the problem is primal feasible then the reported interior-point and basic solutions are a solution to the problem with all integer variables fixed at the value they have in the integer solution. If the problem is primal infeasible, then the solution to the root node problem is reported.

11.5.28 Integer restrictions

MSK_MIO_MODE_IGNORED
The integer constraints are ignored and the problem is solved as a continuous problem.

MSK_MIO_MODE_SATISFIED
Integer restrictions should be satisfied.

11.5.29 Mixed-integer node selection types

MSK_MIO_NODE_SELECTION_FREE
The optimizer decides the node selection strategy.

MSK_MIO_NODE_SELECTION_FIRST
The optimizer employs a depth first node selection strategy.

MSK_MIO_NODE_SELECTION_BEST
The optimizer employs a best bound node selection strategy.

MSK_MIO_NODE_SELECTION_WORST
The optimizer employs a worst bound node selection strategy.

MSK_MIO_NODE_SELECTION_HYBRID
The optimizer employs a hybrid strategy.

MSK_MIO_NODE_SELECTION_PSEUDO
The optimizer employs selects the node based on a pseudo cost estimate.

11.5.30 MPS file format type

MSK_MPS_FORMAT_STRICT
It is assumed that the input file satisfies the MPS format strictly.

MSK_MPS_FORMAT_RELAXED
It is assumed that the input file satisfies a slightly relaxed version of the MPS format.

MSK_MPS_FORMAT_FREE
: It is assumed that the input file satisfies the free MPS format. This implies that spaces are not allowed in names. Otherwise the format is free.

MSK_MPS_FORMAT_CPLEX
: The CPLEX compatible version of the MPS format is employed.

11.5.31 Objective sense types

MSK_OBJECTIVE_SENSE_MINIMIZE
: The problem should be minimized.

MSK_OBJECTIVE_SENSE_MAXIMIZE
: The problem should be maximized.

11.5.32 On/off

MSK_ON
: Switch the option on.

MSK_OFF
: Switch the option off.

11.5.33 Optimizer types

MSK_OPTIMIZER_CONIC
: The optimizer for problems having conic constraints.

MSK_OPTIMIZER_DUAL_SIMPLEX
: The dual simplex optimizer is used.

MSK_OPTIMIZER_FREE
: The optimizer is chosen automatically.

MSK_OPTIMIZER_FREE_SIMPLEX
: One of the simplex optimizers is used.

MSK_OPTIMIZER_INTPNT
: The interior-point optimizer is used.

MSK_OPTIMIZER_MIXED_INT
: The mixed-integer optimizer.

MSK_OPTIMIZER_PRIMAL_SIMPLEX
: The primal simplex optimizer is used.

11.5.34 Ordering strategies

MSK_ORDER_METHOD_FREE
: The ordering method is chosen automatically.

MSK_ORDER_METHOD_APPMINLOC
: Approximate minimum local fill-in ordering is employed.

MSK_ORDER_METHOD_EXPERIMENTAL
: This option should not be used.

MSK_ORDER_METHOD_TRY_GRAPHPAR
: Always try the graph partitioning based ordering.

MSK_ORDER_METHOD_FORCE_GRAPHPAR
Always use the graph partitioning based ordering even if it is worse than the approximate minimum local fill ordering.

MSK_ORDER_METHOD_NONE
No ordering is used.

11.5.35 Presolve method.

MSK_PRESOLVE_MODE_OFF
The problem is not presolved before it is optimized.

MSK_PRESOLVE_MODE_ON
The problem is presolved before it is optimized.

MSK_PRESOLVE_MODE_FREE
It is decided automatically whether to presolve before the problem is optimized.

11.5.36 Parameter type

MSK_PAR_INVALID_TYPE
Not a valid parameter.

MSK_PAR_DOU_TYPE
Is a double parameter.

MSK_PAR_INT_TYPE
Is an integer parameter.

MSK_PAR_STR_TYPE
Is a string parameter.

11.5.37 Problem data items

MSK_PI_VAR
Item is a variable.

MSK_PI_CON
Item is a constraint.

MSK_PI_CONE
Item is a cone.

11.5.38 Problem types

MSK_PROBTYPE_LO
The problem is a linear optimization problem.

MSK_PROBTYPE_QO
The problem is a quadratic optimization problem.

MSK_PROBTYPE_QCQO
The problem is a quadratically constrained optimization problem.

MSK_PROBTYPE_GECO
General convex optimization.

MSK_PROBTYPE_CONIC
A conic optimization.

MSK_PROBTYPE_MIXED
General nonlinear constraints and conic constraints. This combination can not be solved by **MOSEK**.

11.5.39 Problem status keys

MSK_PRO_STA_UNKNOWN
Unknown problem status.

MSK_PRO_STA_PRIM_AND_DUAL_FEAS
The problem is primal and dual feasible.

MSK_PRO_STA_PRIM_FEAS
The problem is primal feasible.

MSK_PRO_STA_DUAL_FEAS
The problem is dual feasible.

MSK_PRO_STA_NEAR_PRIM_AND_DUAL_FEAS
The problem is at least nearly primal and dual feasible.

MSK_PRO_STA_NEAR_PRIM_FEAS
The problem is at least nearly primal feasible.

MSK_PRO_STA_NEAR_DUAL_FEAS
The problem is at least nearly dual feasible.

MSK_PRO_STA_PRIM_INFEAS
The problem is primal infeasible.

MSK_PRO_STA_DUAL_INFEAS
The problem is dual infeasible.

MSK_PRO_STA_PRIM_AND_DUAL_INFEAS
The problem is primal and dual infeasible.

MSK_PRO_STA_ILL_POSED
The problem is ill-posed. For example, it may be primal and dual feasible but have a positive duality gap.

MSK_PRO_STA_PRIM_INFEAS_OR_UNBOUNDED
The problem is either primal infeasible or unbounded. This may occur for mixed-integer problems.

11.5.40 XML writer output mode

MSK_WRITE_XML_MODE_ROW
Write in row order.

MSK_WRITE_XML_MODE_COL
Write in column order.

11.5.41 Response code type

MSK_RESPONSE_OK
The response code is OK.

MSK_RESPONSE_WRN
The response code is a warning.

MSK_RESPONSE_TRM
The response code is an optimizer termination status.

MSK_RESPONSE_ERR
The response code is an error.

MSK_RESPONSE_UNK
The response code does not belong to any class.

11.5.42 Scaling type

MSK_SCALING_FREE
The optimizer chooses the scaling heuristic.

MSK_SCALING_NONE
No scaling is performed.

MSK_SCALING_MODERATE
A conservative scaling is performed.

MSK_SCALING_AGGRESSIVE
A very aggressive scaling is performed.

11.5.43 Scaling method

MSK_SCALING_METHOD_POW2
Scales only with power of 2 leaving the mantissa untouched.

MSK_SCALING_METHOD_FREE
The optimizer chooses the scaling heuristic.

11.5.44 Sensitivity types

MSK_SENSITIVITY_TYPE_BASIS
Basis sensitivity analysis is performed.

MSK_SENSITIVITY_TYPE_OPTIMAL_PARTITION
Optimal partition sensitivity analysis is performed.

11.5.45 Simplex selection strategy

MSK_SIM_SELECTION_FREE
The optimizer chooses the pricing strategy.

MSK_SIM_SELECTION_FULL
The optimizer uses full pricing.

MSK_SIM_SELECTION_ASE
The optimizer uses approximate steepest-edge pricing.

MSK_SIM_SELECTION_DEVEX
The optimizer uses devex steepest-edge pricing (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_SE
The optimizer uses steepest-edge selection (or if it is not available an approximate steep-edge selection).

MSK_SIM_SELECTION_PARTIAL
The optimizer uses a partial selection approach. The approach is usually beneficial if the number of variables is much larger than the number of constraints.

11.5.46 Solution items

MSK_SOL_ITEM_XC
: Solution for the constraints.

MSK_SOL_ITEM_XX
: Variable solution.

MSK_SOL_ITEM_Y
: Lagrange multipliers for equations.

MSK_SOL_ITEM_SLC
: Lagrange multipliers for lower bounds on the constraints.

MSK_SOL_ITEM_SUC
: Lagrange multipliers for upper bounds on the constraints.

MSK_SOL_ITEM_SLX
: Lagrange multipliers for lower bounds on the variables.

MSK_SOL_ITEM_SUX
: Lagrange multipliers for upper bounds on the variables.

MSK_SOL_ITEM_SNX
: Lagrange multipliers corresponding to the conic constraints on the variables.

11.5.47 Solution status keys

MSK_SOL_STA_UNKNOWN
: Status of the solution is unknown.

MSK_SOL_STA_OPTIMAL
: The solution is optimal.

MSK_SOL_STA_PRIM_FEAS
: The solution is primal feasible.

MSK_SOL_STA_DUAL_FEAS
: The solution is dual feasible.

MSK_SOL_STA_PRIM_AND_DUAL_FEAS
: The solution is both primal and dual feasible.

MSK_SOL_STA_NEAR_OPTIMAL
: The solution is nearly optimal.

MSK_SOL_STA_NEAR_PRIM_FEAS
: The solution is nearly primal feasible.

MSK_SOL_STA_NEAR_DUAL_FEAS
: The solution is nearly dual feasible.

MSK_SOL_STA_NEAR_PRIM_AND_DUAL_FEAS
: The solution is nearly both primal and dual feasible.

MSK_SOL_STA_PRIM_INFEAS_CER
: The solution is a certificate of primal infeasibility.

MSK_SOL_STA_DUAL_INFEAS_CER
: The solution is a certificate of dual infeasibility.

MSK_SOL_STA_NEAR_PRIM_INFEAS_CER
: The solution is almost a certificate of primal infeasibility.

MSK_SOL_STA_NEAR_DUAL_INFEAS_CER
: The solution is almost a certificate of dual infeasibility.

MSK_SOL_STA_PRIM_ILLPOSED_CER
The solution is a certificate that the primal problem is illposed.

MSK_SOL_STA_DUAL_ILLPOSED_CER
The solution is a certificate that the dual problem is illposed.

MSK_SOL_STA_INTEGER_OPTIMAL
The primal solution is integer optimal.

MSK_SOL_STA_NEAR_INTEGER_OPTIMAL
The primal solution is near integer optimal.

11.5.48 Solution types

MSK_SOL_BAS
The basic solution.

MSK_SOL_ITR
The interior solution.

MSK_SOL_ITG
The integer solution.

11.5.49 Solve primal or dual form

MSK_SOLVE_FREE
The optimizer is free to solve either the primal or the dual problem.

MSK_SOLVE_PRIMAL
The optimizer should solve the primal problem.

MSK_SOLVE_DUAL
The optimizer should solve the dual problem.

11.5.50 Status keys

MSK_SK_UNK
The status for the constraint or variable is unknown.

MSK_SK_BAS
The constraint or variable is in the basis.

MSK_SK_SUPBAS
The constraint or variable is super basic.

MSK_SK_LOW
The constraint or variable is at its lower bound.

MSK_SK_UPR
The constraint or variable is at its upper bound.

MSK_SK_FIX
The constraint or variable is fixed.

MSK_SK_INF
The constraint or variable is infeasible in the bounds.

11.5.51 Starting point types

MSK_STARTING_POINT_FREE
The starting point is chosen automatically.

MSK_STARTING_POINT_GUESS
The optimizer guesses a starting point.

MSK_STARTING_POINT_CONSTANT
The optimizer constructs a starting point by assigning a constant value to all primal and dual variables. This starting point is normally robust.

MSK_STARTING_POINT_SATISFY_BOUNDS
The starting point is chosen to satisfy all the simple bounds on nonlinear variables. If this starting point is employed, then more care than usual should employed when choosing the bounds on the nonlinear variables. In particular very tight bounds should be avoided.

11.5.52 Stream types

MSK_STREAM_LOG
Log stream. Contains the aggregated contents of all other streams. This means that a message written to any other stream will also be written to this stream.

MSK_STREAM_MSG
Message stream. Log information relating to performance and progress of the optimization is written to this stream.

MSK_STREAM_ERR
Error stream. Error messages are written to this stream.

MSK_STREAM_WRN
Warning stream. Warning messages are written to this stream.

11.5.53 Integer values

MSK_MAX_STR_LEN
Maximum string length allowed in **MOSEK**.

MSK_LICENSE_BUFFER_LENGTH
The length of a license key buffer.

11.5.54 Variable types

MSK_VAR_TYPE_CONT
Is a continuous variable.

MSK_VAR_TYPE_INT
Is an integer variable.

CHAPTER

TWELVE

SUPPORTED FILE FORMATS

MOSEK supports a range of problem and solution formats listed in Table 12.1 and Table 12.2. The **Task format** is **MOSEK**'s native binary format and it supports all features that **MOSEK** supports. The **OPF format** is **MOSEK**'s human-readable alternative that supports nearly all features (everything except semidefinite problems). In general, text formats are significantly slower to read, but can be examined and edited directly in any text editor.

Problem formats

See Table 12.1.

Table 12.1: List of supported file formats for optimization problems.

Format Type	Ext.	Binary/Text	LP	QO	CQO	SDP
LP	lp	plain text	X	X		
MPS	mps	plain text	X	X		
OPF	opf	plain text	X	X	X	
CBF	cbf	plain text	X		X	X
OSiL	xml	xml text	X	X		
Task format	task	binary	X	X	X	X
Jtask format	jtask	text	X	X	X	X

Solution formats

See Table 12.2.

Table 12.2: List of supported solution formats.

Format Type	Ext.	Binary/Text	Description
SOL	sol	plain text	Interior Solution
	bas	plain text	Basic Solution
	int	plain text	Integer
Jsol format	jsol	text	Solution

Compression

MOSEK supports GZIP compression of files. Problem files with an additional `.gz` extension are assumed to be compressed when read, and are automatically compressed when written. For example, a file called

```
problem.mps.gz
```

will be considered as a GZIP compressed MPS file.

12.1 The LP File Format

MOSEK supports the LP file format with some extensions. The LP format is not a completely well-defined standard and hence different optimization packages may interpret the same LP file in slightly different ways. **MOSEK** tries to emulate as closely as possible CPLEX's behavior, but tries to stay backward compatible.

The LP file format can specify problems on the form

$$\begin{array}{lrclcl} \text{minimize/maximize} & & & c^T x + \frac{1}{2} q^o(x) & & \\ \text{subject to} & l^c & \leq & Ax + \frac{1}{2} q(x) & \leq & u^c, \\ & l^x & \leq & x & \leq & u^x, \\ & & & x_{\mathcal{J}} \text{ integer}, & & \end{array}$$

where

- $x \in \mathbb{R}^n$ is the vector of decision variables.
- $c \in \mathbb{R}^n$ is the linear term in the objective.
- $q^o :\in \mathbb{R}^n \rightarrow \mathbb{R}$ is the quadratic term in the objective where

$$q^o(x) = x^T Q^o x$$

and it is assumed that

$$Q^o = (Q^o)^T.$$

- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.
- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.
- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.
- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.
- $q : \mathbb{R}^n \rightarrow \mathbb{R}$ is a vector of quadratic functions. Hence,

$$q_i(x) = x^T Q^i x$$

where it is assumed that

$$Q^i = (Q^i)^T.$$

- $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ is an index set of the integer constrained variables.

12.1.1 File Sections

An LP formatted file contains a number of sections specifying the objective, constraints, variable bounds, and variable types. The section keywords may be any mix of upper and lower case letters.

Objective Function

The first section beginning with one of the keywords

```
max
maximum
maximize
min
minimum
minimize
```

defines the objective sense and the objective function, i.e.

$$c^T x + \frac{1}{2} x^T Q^o x.$$

The objective may be given a name by writing

```
myname:
```

before the expressions. If no name is given, then the objective is named `obj`.

The objective function contains linear and quadratic terms. The linear terms are written as:

```
4 x1 + x2 - 0.1 x3
```

and so forth. The quadratic terms are written in square brackets (`[ ]`) and are either squared or multiplied as in the examples

```
x1^2
```

and

```
x1 * x2
```

There may be zero or more pairs of brackets containing quadratic expressions.

An example of an objective section is

```
minimize
myobj: 4 x1 + x2 - 0.1 x3 + [ x1^2 + 2.1 x1 * x2 ]/2
```

Please note that the quadratic expressions are multiplied with $\frac{1}{2}$, so that the above expression means

$$\text{minimize} \quad 4x_1 + x_2 - 0.1 \cdot x_3 + \tfrac{1}{2}(x_1^2 + 2.1 \cdot x_1 \cdot x_2)$$

If the same variable occurs more than once in the linear part, the coefficients are added, so that `4 x1 + 2 x1` is equivalent to `6 x1`. In the quadratic expressions `x1 * x2` is equivalent to `x2 * x1` and, as in the linear part, if the same variables multiplied or squared occur several times their coefficients are added.

Constraints

The second section beginning with one of the keywords

```
subj to
subject to
s.t.
st
```

defines the linear constraint matrix A and the quadratic matrices Q^i.

A constraint contains a name (optional), expressions adhering to the same rules as in the objective and a bound:

```
subject to
con1: x1 + x2 + [ x3^2 ]/2 <= 5.1
```

The bound type (here <=) may be any of <, <=, =, >, >= (< and <= mean the same), and the bound may be any number.

In the standard LP format it is not possible to define more than one bound, but **MOSEK** supports defining ranged constraints by using double-colon (`::`) instead of a single-colon (`:`) after the constraint name, i.e.

$$-5 \leq x_1 + x_2 \leq 5 \tag{12.1}$$

may be written as

```
con:: -5 < x_1 + x_2 < 5
```

By default **MOSEK** writes ranged constraints this way.

If the files must adhere to the LP standard, ranged constraints must either be split into upper bounded and lower bounded constraints or be written as an equality with a slack variable. For example the expression (12.1) may be written as

$$x_1 + x_2 - sl_1 = 0, \ -5 \leq sl_1 \leq 5.$$

Bounds

Bounds on the variables can be specified in the bound section beginning with one of the keywords

```
bound
bounds
```

The bounds section is optional but should, if present, follow the `subject to` section. All variables listed in the bounds section must occur in either the objective or a constraint.

The default lower and upper bounds are 0 and $+\infty$. A variable may be declared free with the keyword `free`, which means that the lower bound is $-\infty$ and the upper bound is $+\infty$. Furthermore it may be assigned a finite lower and upper bound. The bound definitions for a given variable may be written in one or two lines, and bounds can be any number or $\pm\infty$ (written as `+inf`/`-inf`/`+infinity`/`-infinity`) as in the example

```
bounds
x1 free
x2 <= 5
0.1 <= x2
x3 = 42
2 <= x4 < +inf
```

Variable Types

The final two sections are optional and must begin with one of the keywords

```
bin
binaries
binary
```

and

```
gen
general
```

Under `general` all integer variables are listed, and under `binary` all binary (integer variables with bounds 0 and 1) are listed:

```
general
x1 x2
binary
x3 x4
```

Again, all variables listed in the binary or general sections must occur in either the objective or a constraint.

Terminating Section

Finally, an LP formatted file must be terminated with the keyword

```
end
```

12.1.2 LP File Examples

Linear example lo1.lp

```
\ File: lo1.lp
maximize
obj: 3 x1 + x2 + 5 x3 + x4
subject to
c1:  3 x1 + x2 + 2 x3 = 30
c2:  2 x1 + x2 + 3 x3 + x4 >= 15
c3:  2 x2 + 3 x4 <= 25
bounds
 0 <= x1 <= +infinity
 0 <= x2 <= 10
 0 <= x3 <= +infinity
 0 <= x4 <= +infinity
end
```

Mixed integer example milo1.lp

```
maximize
obj: x1 + 6.4e-01 x2
subject to
c1:  5e+01 x1 + 3.1e+01 x2 <= 2.5e+02
c2:  3e+00 x1 - 2e+00 x2 >= -4e+00
bounds
 0 <= x1 <= +infinity
 0 <= x2 <= +infinity
general
 x1 x2
end
```

12.1.3 LP Format peculiarities

Comments

Anything on a line after a \ is ignored and is treated as a comment.

Names

A name for an objective, a constraint or a variable may contain the letters *a-z*, *A-Z*, the digits *0-9* and the characters

```
!"#$%&()/,.;?@_'`|~
```

The first character in a name must not be a number, a period or the letter `e` or `E`. Keywords must not be used as names.

MOSEK accepts any character as valid for names, except `\0`. A name that is not allowed in LP file will be changed and a warning will be issued.

The algorithm for making names LP valid works as follows: The name is interpreted as an `utf-8` string. For a unicode character `c`:

- If `c==_` (underscore), the output is `__` (two underscores).
- If `c` is a valid LP name character, the output is just `c`.
- If `c` is another character in the ASCII range, the output is `_XX`, where `XX` is the hexadecimal code for the character.
- If `c` is a character in the range *127-65535*, the output is `_uXXXX`, where `XXXX` is the hexadecimal code for the character.
- If `c` is a character above 65535, the output is `_UXXXXXXXX`, where `XXXXXXXX` is the hexadecimal code for the character.

Invalid `utf-8` substrings are escaped as `_XX'`, and if a name starts with a period, *e* or *E*, that character is escaped as `_XX`.

Variable Bounds

Specifying several upper or lower bounds on one variable is possible but **MOSEK** uses only the tightest bounds. If a variable is fixed (with `=`), then it is considered the tightest bound.

MOSEK Extensions to the LP Format

Some optimization software packages employ a more strict definition of the LP format than the one used by **MOSEK**. The limitations imposed by the strict LP format are the following:

- Quadratic terms in the constraints are not allowed.
- Names can be only 16 characters long.
- Lines must not exceed 255 characters in length.

If an LP formatted file created by **MOSEK** should satisfy the strict definition, then the parameter

- *MSK_IPAR_WRITE_LP_STRICT_FORMAT*

should be set; note, however, that some problems cannot be written correctly as a strict LP formatted file. For instance, all names are truncated to 16 characters and hence they may loose their uniqueness and change the problem.

To get around some of the inconveniences converting from other problem formats, **MOSEK** allows lines to contain 1024 characters and names may have any length (shorter than the 1024 characters).

Internally in **MOSEK** names may contain any (printable) character, many of which cannot be used in LP names. Setting the parameters

- *MSK_IPAR_READ_LP_QUOTED_NAMES* and
- *MSK_IPAR_WRITE_LP_QUOTED_NAMES*

allows **MOSEK** to use quoted names. The first parameter tells **MOSEK** to remove quotes from quoted names e.g, `"x1"`, when reading LP formatted files. The second parameter tells **MOSEK** to put quotes around any semi-illegal name (names beginning with a number or a period) and fully illegal name (containing illegal characters). As double quote is a legal character in the LP format, quoting semi-illegal names makes them legal in the pure LP format as long as they are still shorter than 16 characters. Fully illegal names are still illegal in a pure LP file.

12.1.4 The strict LP format

The LP format is not a formal standard and different vendors have slightly different interpretations of the LP format. To make **MOSEK**'s definition of the LP format more compatible with the definitions of other vendors, use the parameter setting

- *MSK_IPAR_WRITE_LP_STRICT_FORMAT* = *MSK_ON*

This setting may lead to truncation of some names and hence to an invalid LP file. The simple solution to this problem is to use the parameter setting

- *MSK_IPAR_WRITE_GENERIC_NAMES* = *MSK_ON*

which will cause all names to be renamed systematically in the output file.

12.1.5 Formatting of an LP File

A few parameters control the visual formatting of LP files written by **MOSEK** in order to make it easier to read the files. These parameters are

- *MSK_IPAR_WRITE_LP_LINE_WIDTH*
- *MSK_IPAR_WRITE_LP_TERMS_PER_LINE*

The first parameter sets the maximum number of characters on a single line. The default value is 80 corresponding roughly to the width of a standard text document.

The second parameter sets the maximum number of terms per line; a term means a sign, a coefficient, and a name (for example `+ 42 elephants`). The default value is 0, meaning that there is no maximum.

Unnamed Constraints

Reading and writing an LP file with **MOSEK** may change it superficially. If an LP file contains unnamed constraints or objective these are given their generic names when the file is read (however unnamed constraints in **MOSEK** are written without names).

12.2 The MPS File Format

MOSEK supports the standard MPS format with some extensions. For a detailed description of the MPS format see the book by Nazareth *[Naz87]*.

12.2.1 MPS File Structure

The version of the MPS format supported by **MOSEK** allows specification of an optimization problem of the form

$$\begin{array}{rcccl} l^c & \leq & Ax + q(x) & \leq & u^c, \\ l^x & \leq & x & \leq & u^x, \\ & & x \in \mathcal{K}, & & \\ & & x_{\mathcal{J}} \quad \text{integer}, & & \end{array} \tag{12.2}$$

where

- $x \in \mathbb{R}^n$ is the vector of decision variables.
- $A \in \mathbb{R}^{m \times n}$ is the constraint matrix.
- $l^c \in \mathbb{R}^m$ is the lower limit on the activity for the constraints.
- $u^c \in \mathbb{R}^m$ is the upper limit on the activity for the constraints.
- $l^x \in \mathbb{R}^n$ is the lower limit on the activity for the variables.
- $u^x \in \mathbb{R}^n$ is the upper limit on the activity for the variables.
- $q : \mathbb{R}^n \rightarrow \mathbb{R}$ is a vector of quadratic functions. Hence,

$$q_i(x) = \frac{1}{2} x^T Q^i x$$

where it is assumed that

$$Q^i = (Q^i)^T.$$

Please note the explicit $\frac{1}{2}$ in the quadratic term and that Q^i is required to be symmetric.

- $\mathcal{K}$ is a convex cone.
- $\mathcal{J} \subseteq \{1, 2, \ldots, n\}$ is an index set of the integer-constrained variables.

An MPS file with one row and one column can be illustrated like this:

```
*        1         2         3         4         5         6
*23456789012345678901234567890123456789012345678901234567890
NAME          [name]
OBJSENSE
[objsense]
OBJNAME
[objname]
ROWS
?  [cname1]
COLUMNS
[vname1]  [cname1]    [value1]     [vname3]  [value2]
RHS
[name]    [cname1]    [value1]     [cname2]  [value2]
RANGES
[name]    [cname1]    [value1]     [cname2]  [value2]
QSECTION      [cname1]
[vname1]  [vname2]    [value1]     [vname3]  [value2]
QMATRIX
[vname1]  [vname2]    [value1]
QUADOBJ
[vname1]  [vname2]    [value1]
QCMATRIX      [cname1]
[vname1]  [vname2]    [value1]
BOUNDS
?? [name]    [vname1]    [value1]
CSECTION      [kname1]    [value1]     [ktype]
[vname1]
ENDATA
```

Here the names in capitals are keywords of the MPS format and names in brackets are custom defined names or values. A couple of notes on the structure:

- Fields: All items surrounded by brackets appear in *fields*. The fields named "`valueN`" are numerical values. Hence, they must have the format

```
  [+|-]XXXXXXX.XXXXXX[[e|E][+|-]XXX]


where
```

```
.. code-block:: text

   X = [0|1|2|3|4|5|6|7|8|9].
```

- Sections: The MPS file consists of several sections where the names in capitals indicate the beginning of a new section. For example, `COLUMNS` denotes the beginning of the columns section.
- Comments: Lines starting with an `*` are comment lines and are ignored by **MOSEK**.
- Keys: The question marks represent keys to be specified later.
- Extensions: The sections `QSECTION` and `CSECTION` are specific **MOSEK** extensions of the MPS format. The sections `QMATRIX`, `QUADOBJ` and `QCMATRIX` are included for sake of compatibility with other vendors extensions to the MPS format.

The standard MPS format is a fixed format, i.e. everything in the MPS file must be within certain fixed positions. **MOSEK** also supports a *free format*. See Sec. 12.2.9 for details.

Linear example `lo1.mps`

A concrete example of a MPS file is presented below:

```
* File: lo1.mps
NAME          lo1
OBJSENSE
    MAX
ROWS
 N  obj
 E  c1
 G  c2
 L  c3
COLUMNS
    x1        obj       3
    x1        c1        3
    x1        c2        2
    x2        obj       1
    x2        c1        1
    x2        c2        1
    x2        c3        2
    x3        obj       5
    x3        c1        2
    x3        c2        3
    x4        obj       1
    x4        c2        1
    x4        c3        3
RHS
    rhs       c1        30
    rhs       c2        15
    rhs       c3        25
RANGES
BOUNDS
 UP bound     x2        10
ENDATA
```

Subsequently each individual section in the MPS format is discussed.

Section `NAME`

In this section a name (`[name]`) is assigned to the problem.

OBJSENSE (optional)

This is an optional section that can be used to specify the sense of the objective function. The OBJSENSE section contains one line at most which can be one of the following

```
MIN
MINIMIZE
MAX
MAXIMIZE
```

It should be obvious what the implication is of each of these four lines.

OBJNAME (optional)

This is an optional section that can be used to specify the name of the row that is used as objective function. The OBJNAME section contains one line at most which has the form

```
objname
```

objname should be a valid row name.

ROWS

A record in the ROWS section has the form

```
?  [cname1]
```

where the requirements for the fields are as follows:

Field	Starting Position	Max Width	required	Description
?	2	1	Yes	Constraint key
[cname1]	5	8	Yes	Constraint name

Hence, in this section each constraint is assigned an unique name denoted by [cname1]. Please note that [cname1] starts in position 5 and the field can be at most 8 characters wide. An initial key ? must be present to specify the type of the constraint. The key can have the values E, G, L, or N with the following interpretation:

Constraint type	l_i^c	u_i^c
E	finite	l_i^c
G	finite	∞
L	$-\infty$	finite
N	$-\infty$	∞

In the MPS format an objective vector is not specified explicitly, but one of the constraints having the key N will be used as the objective vector c . In general, if multiple N type constraints are specified, then the first will be used as the objective vector c.

COLUMNS

In this section the elements of A are specified using one or more records having the form:

```
[vname1]  [cname1]    [value1]     [cname2]  [value2]
```

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

Hence, a record specifies one or two elements a_{ij} of A using the principle that `[vname1]` and `[cname1]` determines j and i respectively. Please note that `[cname1]` must be a constraint name specified in the `ROWS` section. Finally, `[value1]` denotes the numerical value of a_{ij} . Another optional element is specified by `[cname2]`, and `[value2]` for the variable specified by `[vname1]`. Some important comments are:

- All elements belonging to one variable must be grouped together.
- Zero elements of A should not be specified.
- At least one element for each variable should be specified.

`RHS` (optional)

A record in this section has the format

```
[name]     [cname1]     [value1]      [cname2]   [value2]
```

where the requirements for each field are as follows:

Field	Starting Position	Max Width	required	Description
[name]	5	8	Yes	Name of the `RHS` vector
[cname1]	15	8	Yes	Constraint name
[value1]	25	12	Yes	Numerical value
[cname2]	40	8	No	Constraint name
[value2]	50	12	No	Numerical value

The interpretation of a record is that `[name]` is the name of the `RHS` vector to be specified. In general, several vectors can be specified. `[cname1]` denotes a constraint name previously specified in the `ROWS` section. Now, assume that this name has been assigned to the i th constraint and v_1 denotes the value specified by `[value1]`, then the interpretation of v_1 is:

Constraint	l_i^c	u_i^c
type		
E	v_1	v_1
G	v_1	
L		v_1
N		

An optional second element is specified by `[cname2]` and `[value2]` and is interpreted in the same way. Please note that it is not necessary to specify zero elements, because elements are assumed to be zero.

`RANGES` (optional)

A record in this section has the form

```
[name]     [cname1]     [value1]      [cname2]   [value2]
```

where the requirements for each fields are as follows:

Field	Starting Position	Max Width	required	Description
`[name]`	5	8	Yes	Name of the `RANGE` vector
`[cname1]`	15	8	Yes	Constraint name
`[value1]`	25	12	Yes	Numerical value
`[cname2]`	40	8	No	Constraint name
`[value2]`	50	12	No	Numerical value

The records in this section are used to modify the bound vectors for the constraints, i.e. the values in l^c and u^c . A record has the following interpretation: `[name]` is the name of the `RANGE` vector and `[cname1]` is a valid constraint name. Assume that `[cname1]` is assigned to the i th constraint and let v_1 be the value specified by `[value1]`, then a record has the interpretation:

Constraint type	Sign of v_1	l_i^c	u_i^c
E	$-$	$u_i^c + v_1$	
E	$+$		$l_i^c + v_1$
G	$-$ or $+$	$l_i^c + \lvert v_1 \rvert$	
L	$-$ or $+$	$u_i^c - \lvert v_1 \rvert$	
N			

`QSECTION` (optional)

Within the `QSECTION` the label `[cname1]` must be a constraint name previously specified in the `ROWS` section. The label `[cname1]` denotes the constraint to which the quadratic term belongs. A record in the `QSECTION` has the form

```
[vname1]  [vname2]    [value1]     [vname3]  [value2]
```

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
`[vname1]`	5	8	Yes	Variable name
`[vname2]`	15	8	Yes	Variable name
`[value1]`	25	12	Yes	Numerical value
`[vname3]`	40	8	No	Variable name
`[value2]`	50	12	No	Numerical value

A record specifies one or two elements in the lower triangular part of the Q^i matrix where `[cname1]` specifies the i . Hence, if the names `[vname1]` and `[vname2]` have been assigned to the k th and j th variable, then Q^i_{kj} is assigned the value given by `[value1]` An optional second element is specified in the same way by the fields `[vname1]`, `[vname3]`, and `[value2]`.

The example

$$\begin{array}{lccl} \text{minimize} & -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) & & \\ \text{subject to} & x_1 + x_2 + x_3 & \geq & 1, \\ & x \geq 0 & & \end{array}$$

has the following MPS file representation

```
* File: qo1.mps
NAME          qo1
ROWS
N  obj
G  c1
COLUMNS
```

```
x1        c1        1.0
x2        obj       -1.0
x2        c1        1.0
x3        c1        1.0
RHS
rhs       c1        1.0
QSECTION      obj
x1        x1        2.0
x1        x3        -1.0
x2        x2        0.2
x3        x3        2.0
ENDATA
```

Regarding the QSECTIONs please note that:

- Only one QSECTION is allowed for each constraint.
- The QSECTIONs can appear in an arbitrary order after the COLUMNS section.
- All variable names occurring in the QSECTION must already be specified in the COLUMNS section.
- All entries specified in a QSECTION are assumed to belong to the lower triangular part of the quadratic term of Q .

QMATRIX/QUADOBJ (optional)

The QMATRIX and QUADOBJ sections allow to define the quadratic term of the objective function. They differ in how the quadratic term of the objective function is stored:

- QMATRIX It stores all the nonzeros coefficients, withouot taking advantage of the symmetry of the Q matrix.
- QUADOBJ It only store the upper diagonal nonzero elements of the Q matrix.

A record in both sections has the form:

```
[vname1]  [vname2]    [value1]
```

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	Variable name
[vname2]	15	8	Yes	Variable name
[value1]	25	12	Yes	Numerical value

A record specifies one elements of the Q matrix in the objective function . Hence, if the names [vname1] and [vname2] have been assigned to the k th and j th variable, then Q_{kj} is assigned the value given by [value1]. Note that a line must apper for each off-diagonal coefficient if using a QMATRIX section, while only one entry is required in a QUADOBJ section. The quadratic part of the objective function will be evaluated as $1/2x^TQx$.

The example

$$
\begin{array}{lccc}
\text{minimize} & -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) & & \\
\text{subject to} & x_1 + x_2 + x_3 & \geq & 1, \\
 & x \geq 0 & &
\end{array}
$$

has the following MPS file representation using QMATRIX

```
* File: qo1_matrix.mps
NAME          qo1_qmatrix
ROWS
```

```
 N  obj
 G  c1
COLUMNS
    x1        c1        1.0
    x2        obj       -1.0
    x2        c1        1.0
    x3        c1        1.0
RHS
    rhs       c1        1.0
QMATRIX
    x1        x1        2.0
    x1        x3        -1.0
    x3        x1        -1.0
    x2        x2        0.2
    x3        x3        2.0
ENDATA
```

or the following using QUADOBJ

```
* File: qo1_quadobj.mps
NAME          qo1_quadobj
ROWS
 N  obj
 G  c1
COLUMNS
    x1        c1        1.0
    x2        obj       -1.0
    x2        c1        1.0
    x3        c1        1.0
RHS
    rhs       c1        1.0
QUADOBJ
    x1        x1        2.0
    x1        x3        -1.0
    x2        x2        0.2
    x3        x3        2.0
ENDATA
```

Please also note that:

- A `QMATRIX`/`QUADOBJ` section can appear in an arbitrary order after the `COLUMNS` section.
- All variable names occurring in the `QMATRIX`/`QUADOBJ` section must already be specified in the `COLUMNS` section.

12.2.2 `QCMATRIX` (optional)

A `QCMATRIX` section allows to specify the quadratic part of a given constraints. Within the `QCMATRIX` the label `[cname1]` must be a constraint name previously specified in the `ROWS` section. The label `[cname1]` denotes the constraint to which the quadratic term belongs. A record in the `QSECTION` has the form

```
[vname1]  [vname2]    [value1]
```

where the requirements for each field are:

Field	Starting Position	Max Width	required	Description
`[vname1]`	5	8	Yes	Variable name
`[vname2]`	15	8	Yes	Variable name
`[value1]`	25	12	Yes	Numerical value

A record specifies an entry of the Q^i matrix where `[cname1]` specifies the i. Hence, if the names `[vname1]` and `[vname2]` have been assigned to the k th and j th variable, then Q^i_{kj} is assigned the value given by `[value1]`. Moreover, the quadratic term is represented as $1/2x^TQx$.

The example

$$\begin{array}{lccc}
\text{minimize} & x_2 & & \\
\text{subject to} & x_1 + x_2 + x_3 & \geq & 1, \\
& \frac{1}{2}(-2x_1x_3 + 0.2x_2^2 + 2x_3^2) & \leq & 10, \\
& x \geq 0 & &
\end{array}$$

has the following MPS file representation

```
* File: qo1.mps
NAME          qo1
ROWS
 N  obj
 G  c1
 L  q1
COLUMNS
    x1        c1        1.0
    x2        obj       -1.0
    x2        c1        1.0
    x3        c1        1.0
RHS
    rhs       c1        1.0
    rhs       q1        10.0
QCMATRIX      q1
    x1        x1        2.0
    x1        x3        -1.0
    x3        x1        -1.0
    x2        x2        0.2
    x3        x3        2.0
ENDATA
```

Regarding the `QCMATRIX`s please note that:

- Only one `QCMATRIX` is allowed for each constraint.
- The `QCMATRIX`s can appear in an arbitrary order after the `COLUMNS` section.
- All variable names occurring in the `QSECTION` must already be specified in the `COLUMNS` section.
- A `QCMATRIX` does not exploit the symmetry of Q: an off-diagonal entry (i, j) should apperas twice.

12.2.3 `BOUNDS` (optional)

In the `BOUNDS` section changes to the default bounds vectors l^x and u^x are specified. The default bounds vectors are $l^x = 0$ and $u^x = \infty$. Moreover, it is possible to specify several sets of bound vectors. A record in this section has the form

```
?? [name]    [vname1]    [value1]
```

where the requirements for each field are:

Field	Starting Position	Max Width	Required	Description
`??`	2	2	Yes	Bound key
`[name]`	5	8	Yes	Name of the `BOUNDS` vector
`[vname1]`	15	8	Yes	Variable name
`[value1]`	25	12	No	Numerical value

Hence, a record in the BOUNDS section has the following interpretation: [name] is the name of the bound vector and [vname1] is the name of the variable which bounds are modified by the record. ?? and [value1] are used to modify the bound vectors according to the following table:

??	l_j^x	u_j^x	Made integer (added to $\mathcal{J}$)
FR	$-\infty$	∞	No
FX	v_1	v_1	No
LO	v_1	unchanged	No
MI	$-\infty$	unchanged	No
PL	unchanged	∞	No
UP	unchanged	v_1	No
BV	0	1	Yes
LI	$\lceil v_1 \rceil$	unchanged	Yes
UI	unchanged	$\lfloor v_1 \rfloor$	Yes

v_1 is the value specified by [value1].

12.2.4 CSECTION (optional)

The purpose of the CSECTION is to specify the constraint

$$x \in \mathcal{K}.$$

in (12.2). It is assumed that $\mathcal{K}$ satisfies the following requirements. Let

$$x^t \in \mathbb{R}^{n^t}, \quad t = 1, \ldots, k$$

be vectors comprised of parts of the decision variables x so that each decision variable is a member of exactly **one** vector x^t, for example

$$x^1 = \begin{bmatrix} x_1 \\ x_4 \\ x_7 \end{bmatrix} \quad \text{and} \quad x^2 = \begin{bmatrix} x_6 \\ x_5 \\ x_3 \\ x_2 \end{bmatrix}.$$

Next define

$$\mathcal{K} := \left\{ x \in \mathbb{R}^n : \quad x^t \in \mathcal{K}_t, \quad t = 1, \ldots, k \right\}$$

where $\mathcal{K}_t$ must have one of the following forms

- $\mathbb{R}$ set:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} \right\}.$$

- Quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \tag{12.3}$$

- Rotated quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1 x_2 \geq \sum_{j=3}^{n^t} x_j^2, \quad x_1, x_2 \geq 0 \right\}. \tag{12.4}$$

In general, only quadratic and rotated quadratic cones are specified in the MPS file whereas membership of the $\mathbb{R}$ set is not. If a variable is not a member of any other cone then it is assumed to be a member of an $\mathbb{R}$ cone.

Next, let us study an example. Assume that the quadratic cone

$$x_4 \geq \sqrt{x_5^2 + x_8^2}$$

and the rotated quadratic cone

$$x_3 x_7 \geq x_1^2 + x_0^2, \quad x_3, x_7 \geq 0,$$

should be specified in the MPS file. One CSECTION is required for each cone and they are specified as follows:

```
*       1         2         3         4         5         6
*23456789012345678901234567890123456789012345678901234567890
CSECTION      konea       0.0         QUAD
x4
x5
x8
CSECTION      koneb       0.0         RQUAD
x7
x3
x1
x0
```

This first CSECTION specifies the cone (12.3) which is given the name konea. This is a quadratic cone which is specified by the keyword QUAD in the CSECTION header. The 0.0 value in the CSECTION header is not used by the QUAD cone.

The second CSECTION specifies the rotated quadratic cone (12.4). Please note the keyword RQUAD in the CSECTION which is used to specify that the cone is a rotated quadratic cone instead of a quadratic cone. The 0.0 value in the CSECTION header is not used by the RQUAD cone.

In general, a CSECTION header has the format

```
CSECTION      [kname1]    [value1]     [ktype]
```

where the requirement for each field are as follows:

Field	Starting Position	Max Width	Required	Description
[kname1]	5	8	Yes	Name of the cone
[value1]	15	12	No	Cone parameter
[ktype]	25		Yes	Type of the cone.

The possible cone type keys are:

Cone type key	Members	Interpretation.
QUAD	≤ 1	Quadratic cone i.e. (12.3).
RQUAD	≤ 2	Rotated quadratic cone i.e. (12.4).

Please note that a quadratic cone must have at least one member whereas a rotated quadratic cone must have at least two members. A record in the CSECTION has the format

```
[vname1]
```

where the requirements for each field are

Field	Starting Position	Max Width	required	Description
[vname1]	2	8	Yes	A valid variable name

The most important restriction with respect to the CSECTION is that a variable must occur in only one CSECTION.

12.2.5 ENDATA

This keyword denotes the end of the MPS file.

12.2.6 Integer Variables

Using special bound keys in the BOUNDS section it is possible to specify that some or all of the variables should be integer-constrained i.e. be members of $\mathcal{J}$. However, an alternative method is available.

This method is available only for backward compatibility and we recommend that it is not used. This method requires that markers are placed in the COLUMNS section as in the example:

```
COLUMNS
x1        obj       -10.0          c1        0.7
x1        c2        0.5            c3        1.0
x1        c4        0.1
* Start of integer-constrained variables.
MARK000   'MARKER'                 'INTORG'
x2        obj       -9.0           c1        1.0
x2        c2        0.8333333333   c3        0.66666667
x2        c4        0.25
x3        obj       1.0            c6        2.0
MARK001   'MARKER'                 'INTEND'
```

- End of integer-constrained variables.

Please note that special marker lines are used to indicate the start and the end of the integer variables. Furthermore be aware of the following

- IMPORTANT: All variables between the markers are assigned a default lower bound of 0 and a default upper bound of 1. **This may not be what is intended.** If it is not intended, the correct bounds should be defined in the BOUNDS section of the MPS formatted file.
- **MOSEK** ignores field 1, i.e. MARK0001 and MARK001, however, other optimization systems require them.
- Field 2, i.e. MARKER, must be specified including the single quotes. This implies that no row can be assigned the name MARKER.
- Field 3 is ignored and should be left blank.
- Field 4, i.e. INTORG and INTEND, must be specified.
- It is possible to specify several such integer marker sections within the COLUMNS section.

12.2.7 General Limitations

- An MPS file should be an ASCII file.

12.2.8 Interpretation of the MPS Format

Several issues related to the MPS format are not well-defined by the industry standard. However, **MOSEK** uses the following interpretation:

- If a matrix element in the COLUMNS section is specified multiple times, then the multiple entries are added together.

- If a matrix element in a QSECTION section is specified multiple times, then the multiple entries are added together.

12.2.9 The Free MPS Format

MOSEK supports a free format variation of the MPS format. The free format is similar to the MPS file format but less restrictive, e.g. it allows longer names. However, it also presents two main limitations:

- A name must not contain any blanks.
- By default a line in the MPS file must not contain more than 1024 characters. However, by modifying the parameter *MSK_IPAR_READ_MPS_WIDTH* an arbitrary large line width will be accepted.

To use the free MPS format instead of the default MPS format the **MOSEK** parameter *MSK_IPAR_READ_MPS_FORMAT* should be changed.

12.3 The OPF Format

The *Optimization Problem Format (OPF)* is an alternative to LP and MPS files for specifying optimization problems. It is row-oriented, inspired by the CPLEX LP format.

Apart from containing objective, constraints, bounds etc. it may contain complete or partial solutions, comments and extra information relevant for solving the problem. It is designed to be easily read and modified by hand and to be forward compatible with possible future extensions.

Intended use

The OPF file format is meant to replace several other files:

- The LP file format: Any problem that can be written as an LP file can be written as an OPF file too; furthermore it naturally accommodates ranged constraints and variables as well as arbitrary characters in names, fixed expressions in the objective, empty constraints, and conic constraints.
- Parameter files: It is possible to specify integer, double and string parameters along with the problem (or in a separate OPF file).
- Solution files: It is possible to store a full or a partial solution in an OPF file and later reload it.

12.3.1 The File Format

The format uses tags to structure data. A simple example with the basic sections may look like this:

```
[comment]
This is a comment. You may write almost anything here...
[/comment]

# This is a single-line comment.

[objective min 'myobj']
x + 3 y + x^2 + 3 y^2 + z + 1
[/objective]

[constraints]
[con 'con01'] 4 <= x + y  [/con]
[/constraints]

[bounds]
[b] -10 <= x,y <= 10  [/b]
```

```
[cone quad] x,y,z [/cone]
[/bounds]
```

A scope is opened by a tag of the form `[tag]` and closed by a tag of the form `[/tag]`. An opening tag may accept a list of unnamed and named arguments, for examples:

```
[tag value] tag with one unnamed argument [/tag]
[tag arg=value] tag with one named argument in quotes [/tag]
```

Unnamed arguments are identified by their order, while named arguments may appear in any order, but never before an unnamed argument. The `value` can be a quoted, single-quoted or double-quoted text string, i.e.

```
[tag 'value']     single-quoted value [/tag]
[tag arg='value'] single-quoted value [/tag]
[tag "value"]     double-quoted value [/tag]
[tag arg="value"] double-quoted value [/tag]
```

Sections

The recognized tags are

`[comment]`

A comment section. This can contain *almost* any text: Between single quotes (') or double quotes (") any text may appear. Outside quotes the markup characters ([and]) must be prefixed by backslashes. Both single and double quotes may appear alone or inside a pair of quotes if it is prefixed by a backslash.

`[objective]`

The objective function: This accepts one or two parameters, where the first one (in the above example `min`) is either `min` or `max` (regardless of case) and defines the objective sense, and the second one (above `myobj`), if present, is the objective name. The section may contain linear and quadratic expressions. If several objectives are specified, all but the last are ignored.

`[constraints]`

This does not directly contain any data, but may contain the subsection `con` defining a linear constraint.

`[con]` defines a single constraint; if an argument is present (`[con NAME]`) this is used as the name of the constraint, otherwise it is given a null-name. The section contains a constraint definition written as linear and quadratic expressions with a lower bound, an upper bound, with both or with an equality. Examples:

```
[constraints]
[con 'con1'] 0 <= x + y       [/con]
[con 'con2'] 0 >= x + y       [/con]
[con 'con3'] 0 <= x + y <= 10 [/con]
[con 'con4']      x + y  = 10 [/con]
[/constraints]
```

Constraint names are unique. If a constraint is specified which has the same name as a previously defined constraint, the new constraint replaces the existing one.

[bounds]

This does not directly contain any data, but may contain the subsections `b` (linear bounds on variables) and `cone` (quadratic cone).

`[b]`. Bound definition on one or several variables separated by comma (`,`). An upper or lower bound on a variable replaces any earlier defined bound on that variable. If only one bound (upper or lower) is given only this bound is replaced. This means that upper and lower bounds can be specified separately. So the OPF bound definition:

```
[b]  x,y >= -10  [/b]
[b]  x,y <= 10   [/b]
```

results in the bound $-10 \leq x, y \leq 10$.

`[cone]`. currently supports the *quadratic cone* and the *rotated quadratic cone*.

A conic constraint is defined as a set of variables which belong to a single unique cone.

- A quadratic cone of n variables $x_1, \ldots, x_n$ defines a constraint of the form

$$x_1^2 \geq \sum_{i=2}^{n} x_i^2, \quad x_1 \geq 0.$$

- A rotated quadratic cone of n variables $x_1, \ldots, x_n$ defines a constraint of the form

$$2x_1 x_2 \geq \sum_{i=3}^{n} x_i^2, \quad x_1, x_2 \geq 0.$$

A `[bounds]`-section example:

```
[bounds]
[b]  0 <= x,y <= 10  [/b] # ranged bound
[b] 10 >= x,y >=  0  [/b] # ranged bound
[b]  0 <= x,y <= inf [/b] # using inf
[b]       x,y free   [/b] # free variables
# Let (x,y,z,w) belong to the cone K
[cone quad]  x,y,z,w  [/cone] # quadratic cone
[cone rquad] x,y,z,w  [/cone] # rotated quadratic cone
[/bounds]
```

By default all variables are free.

[variables]

This defines an ordering of variables as they should appear in the problem. This is simply a space-separated list of variable names. Optionally, an attribute can be added `[variables disallow_new_variables]` indicating that if any variable not listed here occurs later in the file it is an error.

[integer]

This contains a space-separated list of variables and defines the constraint that the listed variables must be integer values.

[hints]

This may contain only non-essential data; for example estimates of the number of variables, constraints and non-zeros. Placed before all other sections containing data this may reduce the time spent reading the file.

In the `hints` section, any subsection which is not recognized by **MOSEK** is simply ignored. In this section a hint in a subsection is defined as follows:

```
[hint ITEM] value [/hint]
```

where `ITEM` may be replaced by `numvar` (number of variables), `numcon` (number of linear/quadratic constraints), `numanz` (number of linear non-zeros in constraints) and `numqnz` (number of quadratic non-zeros in constraints).

[solutions]

This section can contain a set of full or partial solutions to a problem. Each solution must be specified using a `[solution]`-section, i.e.

```
[solutions]
[solution]...[/solution] #solution 1
[solution]...[/solution] #solution 2
#other solutions....
[solution]...[/solution] #solution n
[/solutions]
```

Note that a `[solution]`-section must be always specified inside a `[solutions]`-section. The syntax of a `[solution]`-section is the following:

```
[solution SOLTYPE status=STATUS]...[/solution]
```

where `SOLTYPE` is one of the strings

- `interior`, a non-basic solution,
- `basic`, a basic solution,
- `integer`, an integer solution,

and `STATUS` is one of the strings

- `UNKNOWN`,
- `OPTIMAL`,
- `INTEGER_OPTIMAL`,
- `PRIM_FEAS`,
- `DUAL_FEAS`,
- `PRIM_AND_DUAL_FEAS`,
- `NEAR_OPTIMAL`,
- `NEAR_PRIM_FEAS`,
- `NEAR_DUAL_FEAS`,
- `NEAR_PRIM_AND_DUAL_FEAS`,
- `PRIM_INFEAS_CER`,
- `DUAL_INFEAS_CER`,
- `NEAR_PRIM_INFEAS_CER`,

- NEAR_DUAL_INFEAS_CER,
- NEAR_INTEGER_OPTIMAL.

Most of these values are irrelevant for input solutions; when constructing a solution for simplex hot-start or an initial solution for a mixed integer problem the safe setting is UNKNOWN.

A [solution]-section contains [con] and [var] sections. Each [con] and [var] section defines solution information for a single variable or constraint, specified as list of KEYWORD/value pairs, in any order, written as

```
KEYWORD=value
```

Allowed keywords are as follows:

- sk. The status of the item, where the value is one of the following strings:
  - LOW, the item is on its lower bound.
  - UPR, the item is on its upper bound.
  - FIX, it is a fixed item.
  - BAS, the item is in the basis.
  - SUPBAS, the item is super basic.
  - UNK, the status is unknown.
  - INF, the item is outside its bounds (infeasible).
- lvl Defines the level of the item.
- sl Defines the level of the dual variable associated with its lower bound.
- su Defines the level of the dual variable associated with its upper bound.
- sn Defines the level of the variable associated with its cone.
- y Defines the level of the corresponding dual variable (for constraints only).

A [var] section should always contain the items sk, lvl, sl and su. Items sl and su are not required for integer solutions.

A [con] section should always contain sk, lvl, sl, su and y.

An example of a solution section

```
[solution basic status=UNKNOWN]
[var x0] sk=LOW    lvl=5.0      [/var]
[var x1] sk=UPR    lvl=10.0     [/var]
[var x2] sk=SUPBAS lvl=2.0  sl=1.5 su=0.0 [/var]

[con c0] sk=LOW    lvl=3.0 y=0.0 [/con]
[con c0] sk=UPR    lvl=0.0 y=5.0 [/con]
[/solution]
```

- [vendor] This contains solver/vendor specific data. It accepts one argument, which is a vendor ID – for **MOSEK** the ID is simply mosek – and the section contains the subsection parameters defining solver parameters. When reading a vendor section, any unknown vendor can be safely ignored. This is described later.

Comments using the # may appear anywhere in the file. Between the # and the following line-break any text may be written, including markup characters.

Numbers

Numbers, when used for parameter values or coefficients, are written in the usual way by the `printf` function. That is, they may be prefixed by a sign (`+` or `-`) and may contain an integer part, decimal part and an exponent. The decimal point is always `.` (a dot). Some examples are

```
1
1.0
.0
1.
1e10
1e+10
1e-10
```

Some *invalid* examples are

```
e10   # invalid, must contain either integer or decimal part
.     # invalid
.e10  # invalid
```

More formally, the following standard regular expression describes numbers as used:

```
[+|-]?([0-9]+[.][0-9]*|[.][0-9]+)([eE][+|-]?[0-9]+)?
```

Names

Variable names, constraint names and objective name may contain arbitrary characters, which in some cases must be enclosed by quotes (single or double) that in turn must be preceded by a backslash. Unquoted names must begin with a letter (`a-z` or `A-Z`) and contain only the following characters: the letters `a-z` and `A-Z`, the digits `0-9`, braces (`{` and `}`) and underscore (`_`).

Some examples of legal names:

```
an_unquoted_name
another_name{123}
'single quoted name'
"double quoted name"
"name with \\"quote\\" in it"
"name with []s in it"
```

12.3.2 Parameters Section

In the `vendor` section solver parameters are defined inside the `parameters` subsection. Each parameter is written as

```
[p PARAMETER_NAME] value [/p]
```

where `PARAMETER_NAME` is replaced by a **MOSEK** parameter name, usually of the form `MSK_IPAR_...`, `MSK_DPAR_...` or `MSK_SPAR_...`, and the `value` is replaced by the value of that parameter; both integer values and named values may be used. Some simple examples are

```
[vendor mosek]
[parameters]
[p MSK_IPAR_OPF_MAX_TERMS_PER_LINE] 10     [/p]
[p MSK_IPAR_OPF_WRITE_PARAMETERS]   MSK_ON [/p]
[p MSK_DPAR_DATA_TOL_BOUND_INF]     1.0e18 [/p]
[/parameters]
[/vendor]
```

12.3.3 Writing OPF Files from MOSEK

To write an OPF file set the parameter *MSK_IPAR_WRITE_DATA_FORMAT* to *MSK_DATA_FORMAT_OP* as this ensures that OPF format is used.

Then modify the following parameters to define what the file should contain:

MSK_IPAR_OPF_WRITE_SOL_BAS	Include basic solution, if defined.
MSK_IPAR_OPF_WRITE_SOL_ITG	Include integer solution, if defined.
MSK_IPAR_OPF_WRITE_SOL_ITR	Include interior solution, if defined.
MSK_IPAR_OPF_WRITE_SOLUTIONS	Include solutions if they are defined. If this is off, no solutions are included.
MSK_IPAR_OPF_WRITE_HEADER	Include a small header with comments.
MSK_IPAR_OPF_WRITE_PROBLEM	Include the problem itself — objective, constraints and bounds.
MSK_IPAR_OPF_WRITE_PARAMETERS	Include all parameter settings.
MSK_IPAR_OPF_WRITE_HINTS	Include hints about the size of the problem.

12.3.4 Examples

This section contains a set of small examples written in OPF and describing how to formulate linear, quadratic and conic problems.

Linear Example lo1.opf

Consider the example:

$$
\begin{array}{lccccccccc}
\text{maximize} & 3x_0 & + & 1x_1 & + & 5x_2 & + & 1x_3 & & \\
\text{subject to} & 3x_0 & + & 1x_1 & + & 2x_2 & & & = & 30, \\
& 2x_0 & + & 1x_1 & + & 3x_2 & + & 1x_3 & \geq & 15, \\
& & & 2x_1 & & & + & 3x_3 & \leq & 25,
\end{array}
$$

having the bounds

$$
\begin{array}{ccccc}
0 & \leq & x_0 & \leq & \infty, \\
0 & \leq & x_1 & \leq & 10, \\
0 & \leq & x_2 & \leq & \infty, \\
0 & \leq & x_3 & \leq & \infty.
\end{array}
$$

In the OPF format the example is displayed as shown in Listing 12.1.

Listing 12.1: Example of an OPF file for a linear problem.

```
[comment]
  The lo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 4 [/hint]
  [hint NUMCON] 3 [/hint]
  [hint NUMANZ] 9 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3 x4
[/variables]

[objective maximize 'obj']
   3 x1 + x2 + 5 x3 + x4
[/objective]
```

```
[constraints]
  [con 'c1'] 3 x1 +   x2 + 2 x3         = 30 [/con]
  [con 'c2'] 2 x1 +   x2 + 3 x3 +   x4 >= 15 [/con]
  [con 'c3']        2 x2        + 3 x4 <= 25 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
  [b] 0 <= x2 <= 10 [/b]
[/bounds]
```

Quadratic Example `qo1.opf`

An example of a quadratic optimization problem is

$$
\begin{array}{lccl}
\text{minimize} & & & x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\
\text{subject to} & 1 & \leq & x_1 + x_2 + x_3, \\
 & & & x \geq 0.
\end{array}
$$

This can be formulated in `opf` as shown below.

Listing 12.2: Example of an OPF file for a quadratic problem.

```
[comment]
  The qo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 3 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 3 [/hint]
  [hint NUMQNZ] 4 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3
[/variables]

[objective minimize 'obj']
  # The quadratic terms are often written with a factor of 1/2 as here,
  # but this is not required.

   - x2 + 0.5 ( 2.0 x1 ^ 2 - 2.0 x3 * x1 + 0.2 x2 ^ 2 + 2.0 x3 ^ 2 )
[/objective]

[constraints]
  [con 'c1'] 1.0 <= x1 + x2 + x3 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
[/bounds]
```

Conic Quadratic Example `cqo1.opf`

Consider the example:

$$\begin{array}{lccl} \text{minimize} & x_3 + x_4 + x_5 & & \\ \text{subject to} & x_0 + x_1 + 2x_2 & = & 1, \\ & x_0, x_1, x_2 & \geq & 0, \\ & x_3 \geq \sqrt{x_0^2 + x_1^2}, & & \\ & 2x_4 x_5 \geq x_2^2. & & \end{array}$$

Please note that the type of the cones is defined by the parameter to `[cone ...]`; the content of the `cone`-section is the names of variables that belong to the cone. The resulting OPF file is in Listing 12.3.

Listing 12.3: Example of an OPF file for a conic quadratic problem.

```
[comment]
  The cqo1 example in OPF format.
[/comment]

[hints]
  [hint NUMVAR] 6 [/hint]
  [hint NUMCON] 1 [/hint]
  [hint NUMANZ] 3 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2 x3 x4 x5 x6
[/variables]

[objective minimize 'obj']
   x4 + x5 + x6
[/objective]

[constraints]
  [con 'c1']  x1 + x2 + 2e+00 x3 = 1e+00 [/con]
[/constraints]

[bounds]
  # We let all variables default to the positive orthant
  [b] 0 <= * [/b]

  # ...and change those that differ from the default
  [b] x4,x5,x6 free [/b]

  # Define quadratic cone: x4 >= sqrt( x1^2 + x2^2 )
  [cone quad 'k1'] x4, x1, x2 [/cone]

  # Define rotated quadratic cone: 2 x5 x6 >= x3^2
  [cone rquad 'k2'] x5, x6, x3 [/cone]
[/bounds]
```

Mixed Integer Example `milo1.opf`

Consider the mixed integer problem:

$$\begin{array}{lccl} \text{maximize} & x_0 + 0.64x_1 & & \\ \text{subject to} & 50x_0 + 31x_1 & \leq & 250, \\ & 3x_0 - 2x_1 & \geq & -4, \\ & x_0, x_1 \geq 0 & & \text{and integer} \end{array}$$

This can be implemented in `OPF` with the file in Listing 12.4.

Listing 12.4: Example of an OPF file for a mixed-integer linear problem.

```
[comment]
  The milo1 example in OPF format
[/comment]

[hints]
  [hint NUMVAR] 2 [/hint]
  [hint NUMCON] 2 [/hint]
  [hint NUMANZ] 4 [/hint]
[/hints]

[variables disallow_new_variables]
  x1 x2
[/variables]

[objective maximize 'obj']
   x1 + 6.4e-1 x2
[/objective]

[constraints]
  [con 'c1'] 5e+1 x1 + 3.1e+1 x2 <= 2.5e+2 [/con]
  [con 'c2'] -4 <= 3 x1 - 2 x2 [/con]
[/constraints]

[bounds]
  [b] 0 <= * [/b]
[/bounds]

[integer]
  x1 x2
[/integer]
```

12.4 The CBF Format

This document constitutes the technical reference manual of the *Conic Benchmark Format* with file extension: `.cbf` or `.CBF`. It unifies linear, second-order cone (also known as conic quadratic) and semidefinite optimization with mixed-integer variables. The format has been designed with benchmark libraries in mind, and therefore focuses on compact and easily parsable representations. The problem structure is separated from the problem data, and the format moreover facilitates benchmarking of hotstart capability through sequences of changes.

12.4.1 How Instances Are Specified

This section defines the spectrum of conic optimization problems that can be formulated in terms of the keywords of the CBF format.

In the CBF format, conic optimization problems are considered in the following form:

$$
\begin{array}{lll}
\min / \max & g^{obj} & \\
 & g_i \in \mathcal{K}_i, & i \in \mathcal{I}, \\
\text{s.t.} & G_i \in \mathcal{K}_i, & i \in \mathcal{I}^{PSD}, \\
 & x_j \in \mathcal{K}_j, & j \in \mathcal{J}, \\
 & \overline{X}_j \in \mathcal{K}_j, & j \in \mathcal{J}^{PSD}.
\end{array}
\tag{12.5}
$$

- **Variables** are either scalar variables, x_j for $j \in \mathcal{J}$, or variables, $\overline{X}_j$ for $j \in \mathcal{J}^{PSD}$. Scalar variables can also be declared as integer.

- **Constraints** are affine expressions of the variables, either scalar-valued g_i for $i \in \mathcal{I}$, or matrix-valued G_i for $i \in \mathcal{I}^{PSD}$

$$g_i = \sum_{j \in \mathcal{J}^{PSD}} \langle F_{ij}, X_j \rangle + \sum_{j \in \mathcal{J}} a_{ij} x_j + b_i,$$
$$G_i = \sum_{j \in \mathcal{J}} x_j H_{ij} + D_i.$$

- The **objective function** is a scalar-valued affine expression of the variables, either to be minimized or maximized. We refer to this expression as g^{obj}

$$g^{obj} = \sum_{j \in \mathcal{J}^{PSD}} \langle F_j^{obj}, X_j \rangle + \sum_{j \in \mathcal{J}} a_j^{obj} x_j + b^{obj}.$$

CBF format can represent the following cones $\mathcal{K}$:

- **Free domain** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n\}, \text{ for } n \geq 1.$$

- **Positive orthant** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j \geq 0 \text{ for } j = 1, \ldots, n\}, \text{ for } n \geq 1.$$

- **Negative orthant** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j \leq 0 \text{ for } j = 1, \ldots, n\}, \text{ for } n \geq 1.$$

- **Fixpoint zero** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j = 0 \text{ for } j = 1, \ldots, n\}, \text{ for } n \geq 1.$$

- **Quadratic cone** - A cone in the second-order cone family defined by

$$\left\{ \begin{pmatrix} p \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R}^{n-1},\ p^2 \geq x^T x,\ p \geq 0 \right\}, \text{ for } n \geq 2.$$

- **Rotated quadratic cone** - A cone in the second-order cone family defined by

$$\left\{ \begin{pmatrix} p \\ q \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^{n-2},\ 2pq \geq x^T x,\ p \geq 0,\ q \geq 0 \right\}, \text{ for } n \geq 3.$$

12.4.2 The Structure of CBF Files

This section defines how information is written in the CBF format, without being specific about the type of information being communicated.

All information items belong to exactly one of the three groups of information. These information groups, and the order they must appear in, are:

1. File format.
2. Problem structure.
3. Problem data.

The first group, file format, provides information on how to interpret the file. The second group, problem structure, provides the information needed to deduce the type and size of the problem instance. Finally, the third group, problem data, specifies the coefficients and constants of the problem instance.

Information items

The format is composed as a list of information items. The first line of an information item is the `KEYWORD`, revealing the type of information provided. The second line - of some keywords only - is the `HEADER`, typically revealing the size of information that follows. The remaining lines are the `BODY` holding the actual information to be specified.

```
KEYWORD
BODY

KEYWORD
HEADER
BODY
```

The `KEYWORD` determines how each line in the `HEADER` and `BODY` is structured. Moreover, the number of lines in the `BODY` follows either from the `KEYWORD`, the `HEADER`, or from another information item required to precede it.

Embedded hotstart-sequences

A sequence of problem instances, based on the same problem structure, is within a single file. This is facilitated via the `CHANGE` within the problem data information group, as a separator between the information items of each instance. The information items following a `CHANGE` keyword are appending to, or changing (e.g., setting coefficients back to their default value of zero), the problem data of the preceding instance.

The sequence is intended for benchmarking of hotstart capability, where the solvers can reuse their internal state and solution (subject to the achieved accuracy) as warmpoint for the succeeding instance. Whenever this feature is unsupported or undesired, the keyword `CHANGE` should be interpreted as the end of file.

File encoding and line width restrictions

The format is based on the US-ASCII printable character set with two extensions as listed below. Note, by definition, that none of these extensions can be misinterpreted as printable US-ASCII characters:

- A line feed marks the end of a line, carriage returns are ignored.
- Comment-lines may contain unicode characters in UTF-8 encoding.

The line width is restricted to 512 bytes, with 3 bytes reserved for the potential carriage return, line feed and null-terminator.

Integers and floating point numbers must follow the ISO C decimal string representation in the standard C locale. The format does not impose restrictions on the magnitude of, or number of significant digits in numeric data, but the use of 64-bit integers and 64-bit IEEE 754 floating point numbers should be sufficient to avoid loss of precision.

Comment-line and whitespace rules

The format allows single-line comments respecting the following rule:

- Lines having first byte equal to '#' (US-ASCII 35) are comments, and should be ignored. Comments are only allowed between information items.

Given that a line is not a comment-line, whitespace characters should be handled according to the following rules:

- Leading and trailing whitespace characters should be ignored.
  - The seperator between multiple pieces of information on one line, is either one or more whitespace characters.
- Lines containing only whitespace characters are empty, and should be ignored. Empty lines are only allowed between information items.

12.4.3 Problem Specification

The problem structure

The problem structure defines the objective sense, whether it is minimization and maximization. It also defines the index sets, $\mathcal{J}$, $\mathcal{J}^{PSD}$, $\mathcal{I}$ and $\mathcal{I}^{PSD}$, which are all numbered from zero, $\{0, 1, \ldots\}$, and empty until explicitly constructed.

- **Scalar variables** are constructed in vectors restricted to a conic domain, such as $(x_0, x_1) \in \mathbb{R}^2_+$, $(x_2, x_3, x_4) \in \mathcal{Q}^3$, etc. In terms of the Cartesian product, this generalizes to

$$x \in \mathcal{K}_1^{n_1} \times \mathcal{K}_2^{n_2} \times \cdots \times \mathcal{K}_k^{n_k}$$

  which in the CBF format becomes:

```
VAR
n k
K1 n1
K2 n2
...
Kk nk
```

  where $\sum_i n_i = n$ is the total number of scalar variables. The list of supported cones is found in Table 12.3. Integrality of scalar variables can be specified afterwards.

- **PSD variables** are constructed one-by-one. That is, $X_j \succeq \mathbf{0}^{n_j \times n_j}$ for $j \in \mathcal{J}^{PSD}$, constructs a matrix-valued variable of size $n_j \times n_j$ restricted to be symmetric positive semidefinite. In the CBF format, this list of constructions becomes:

```
PSDVAR
N
n1
n2
...
nN
```

  where N is the total number of PSD variables.

- **Scalar constraints** are constructed in vectors restricted to a conic domain, such as $(g_0, g_1) \in \mathbb{R}^2_+$, $(g_2, g_3, g_4) \in \mathcal{Q}^3$, etc. In terms of the Cartesian product, this generalizes to

$$g \in \mathcal{K}_1^{m_1} \times \mathcal{K}_2^{m_2} \times \cdots \times \mathcal{K}_k^{m_k}$$

  which in the CBF format becomes:

```
CON
m k
K1 m1
K2 m2
..
Kk mk
```

where $\sum_i m_i = m$ is the total number of scalar constraints. The list of supported cones is found in Table 12.3.

- **PSD constraints** are constructed one-by-one. That is, $G_i \succeq \mathbf{0}^{m_i \times m_i}$ for $i \in \mathcal{I}^{PSD}$, constructs a matrix-valued affine expressions of size $m_i \times m_i$ restricted to be symmetric positive semidefinite. In the CBF format, this list of constructions becomes

```
PSDCON
M
m1
m2
..
mM
```

where M is the total number of PSD constraints.

With the objective sense, variables (with integer indications) and constraints, the definitions of the many affine expressions follow in problem data.

Problem data

The problem data defines the coefficients and constants of the affine expressions of the problem instance. These are considered zero until explicitly defined, implying that instances with no keywords from this information group are, in fact, valid. Duplicating or conflicting information is a failure to comply with the standard. Consequently, two coefficients written to the same position in a matrix (or to transposed positions in a symmetric matrix) is an error.

The affine expressions of the objective, g^{obj}, of the scalar constraints, g_i, and of the PSD constraints, G_i, are defined separately. The following notation uses the standard trace inner product for matrices, $\langle X, Y \rangle = \sum_{i,j} X_{ij} Y_{ij}$.

- The affine expression of the objective is defined as

$$g^{obj} = \sum_{j \in \mathcal{J}^{PSD}} \langle F_j^{obj}, X_j \rangle + \sum_{j \in \mathcal{J}} a_j^{obj} x_j + b^{obj},$$

in terms of the symmetric matrices, F_j^{obj}, and scalars, a_j^{obj} and b^{obj}.

- The affine expressions of the scalar constraints are defined, for $i \in \mathcal{I}$, as

$$g_i = \sum_{j \in \mathcal{J}^{PSD}} \langle F_{ij}, X_j \rangle + \sum_{j \in \mathcal{J}} a_{ij} x_j + b_i,$$

in terms of the symmetric matrices, F_{ij}, and scalars, a_{ij} and b_i.

- The affine expressions of the PSD constraints are defined, for $i \in \mathcal{I}^{PSD}$, as

$$G_i = \sum_{j \in \mathcal{J}} x_j H_{ij} + D_i,$$

in terms of the symmetric matrices, H_{ij} and D_i.

List of cones

The format uses an explicit syntax for symmetric positive semidefinite cones as shown above. For scalar variables and constraints, constructed in vectors, the supported conic domains and their minimum sizes are given as follows.

Table 12.3: Cones available in the CBF format

Name	CBF keyword	Cone family
Free domain	`F`	linear
Positive orthant	`L+`	linear
Negative orthant	`L-`	linear
Fixpoint zero	`L=`	linear
Quadratic cone	`Q`	second-order
Rotated quadratic cone	`QR`	second-order

12.4.4 File Format Keywords

VER

Description: The version of the Conic Benchmark Format used to write the file.

`HEADER`: None

`BODY`: One line formatted as:

```
INT
```

This is the version number.

Must appear exactly once in a file, as the first keyword.

OBJSENSE

Description: Define the objective sense.

`HEADER`: None

`BODY`: One line formatted as:

```
STR
```

having `MIN` indicates minimize, and `MAX` indicates maximize. Capital letters are required.

Must appear exactly once in a file.

PSDVAR

Description: Construct the PSD variables.

`HEADER`: One line formatted as:

```
INT
```

This is the number of PSD variables in the problem.

`BODY`: A list of lines formatted as:

```
INT
```

This indicates the number of rows (equal to the number of columns) in the matrix-valued PSD variable. The number of lines should match the number stated in the header.

VAR

Description: Construct the scalar variables.

HEADER: One line formatted as:

```
INT INT
```

This is the number of scalar variables, followed by the number of conic domains they are restricted to.

BODY: A list of lines formatted as:

```
STR INT
```

This indicates the cone name (see Table 12.3), and the number of scalar variables restricted to this cone. These numbers should add up to the number of scalar variables stated first in the header. The number of lines should match the second number stated in the header.

INT

Description: Declare integer requirements on a selected subset of scalar variables.

HEADER: one line formatted as:

```
INT
```

This is the number of integer scalar variables in the problem.

BODY: a list of lines formatted as:

```
INT
```

This indicates the scalar variable index $j \in \mathcal{J}$. The number of lines should match the number stated in the header.

Can only be used after the keyword VAR.

PSDCON

Description: Construct the PSD constraints.

HEADER: One line formatted as:

```
INT
```

This is the number of PSD constraints in the problem.

BODY: A list of lines formatted as:

```
INT
```

This indicates the number of rows (equal to the number of columns) in the matrix-valued affine expression of the PSD constraint. The number of lines should match the number stated in the header.

Can only be used after these keywords: PSDVAR, VAR.

CON

Description: Construct the scalar constraints.

HEADER: One line formatted as:

```
INT INT
```

This is the number of scalar constraints, followed by the number of conic domains they restrict to.

BODY: A list of lines formatted as:

```
STR INT
```

This indicates the cone name (see Table 12.3), and the number of affine expressions restricted to this cone. These numbers should add up to the number of scalar constraints stated first in the header. The number of lines should match the second number stated in the header.

Can only be used after these keywords: PSDVAR, VAR

OBJFCOORD

Description: Input sparse coordinates (quadruplets) to define the symmetric matrices F_j^{obj}, as used in the objective.

HEADER: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

```
INT INT INT REAL
```

This indicates the PSD variable index $j \in \mathcal{J}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

OBJACOORD

Description: Input sparse coordinates (pairs) to define the scalars, a_j^{obj}, as used in the objective.

HEADER: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

```
INT REAL
```

This indicates the scalar variable index $j \in \mathcal{J}$ and the coefficient value. The number of lines should match the number stated in the header.

OBJBCOORD

Description: Input the scalar, b^{obj}, as used in the objective.

HEADER: None.

BODY: One line formatted as:

```
REAL
```

This indicates the coefficient value.

FCOORD

Description: Input sparse coordinates (quintuplets) to define the symmetric matrices, F_{ij}, as used in the scalar constraints.

HEADER: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

```
INT INT INT INT REAL
```

This indicates the scalar constraint index $i \in \mathcal{I}$, the PSD variable index $j \in \mathcal{J}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

ACOORD

Description: Input sparse coordinates (triplets) to define the scalars, a_{ij}, as used in the scalar constraints.

HEADER: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

```
INT INT REAL
```

This indicates the scalar constraint index $i \in \mathcal{I}$, the scalar variable index $j \in \mathcal{J}$ and the coefficient value. The number of lines should match the number stated in the header.

BCOORD

Description: Input sparse coordinates (pairs) to define the scalars, b_i, as used in the scalar constraints.

HEADER: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

BODY: A list of lines formatted as:

```
INT REAL
```

This indicates the scalar constraint index $i \in \mathcal{I}$ and the coefficient value. The number of lines should match the number stated in the header.

HCOORD

Description: Input sparse coordinates (quintuplets) to define the symmetric matrices, H_{ij}, as used in the PSD constraints.

`HEADER`: One line formatted as:

```
INT
```

This is the number of coordinates to be specified.

`BODY`: A list of lines formatted as

```
INT INT INT INT REAL
```

This indicates the PSD constraint index $i \in \mathcal{I}^{PSD}$, the scalar variable index $j \in \mathcal{J}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

DCOORD

Description: Input sparse coordinates (quadruplets) to define the symmetric matrices, D_i, as used in the PSD constraints.

`HEADER`: One line formatted as

```
INT
```

This is the number of coordinates to be specified.

`BODY`: A list of lines formatted as:

```
INT INT INT REAL
```

This indicates the PSD constraint index $i \in \mathcal{I}^{PSD}$, the row index, the column index and the coefficient value. The number of lines should match the number stated in the header.

CHANGE

Start of a new instance specification based on changes to the previous. Can be interpreted as the end of file when the hotstart-sequence is unsupported or undesired.

`BODY`: None

`Header`: None

12.4.5 CBF Format Examples

Minimal Working Example

The conic optimization problem (12.6) , has three variables in a quadratic cone - first one is integer - and an affine expression in domain 0 (equality constraint).

$$
\begin{array}{ll}
\text{minimize} & 5.1\, x_0 \\
\text{subject to} & 6.2\, x_1 + 7.3\, x_2 - 8.4 \in \{0\} \\
 & x \in \mathcal{Q}^3,\ x_0 \in \mathbb{Z}.
\end{array} \tag{12.6}
$$

Its formulation in the Conic Benchmark Format begins with the version of the CBF format used, to safeguard against later revisions.

```
VER
1
```

Next follows the problem structure, consisting of the objective sense, the number and domain of variables, the indices of integer variables, and the number and domain of scalar-valued affine expressions (i.e., the equality constraint).

```
OBJSENSE
MIN

VAR
3 1
Q 3

INT
1
0

CON
1 1
L= 1
```

Finally follows the problem data, consisting of the coefficients of the objective, the coefficients of the constraints, and the constant terms of the constraints. All data is specified on a sparse coordinate form.

```
OBJACOORD
1
0 5.1

ACOORD
2
0 1 6.2
0 2 7.3

BCOORD
1
0 -8.4
```

This concludes the example.

Mixing Linear, Second-order and Semidefinite Cones

The conic optimization problem (12.7), has a semidefinite cone, a quadratic cone over unordered subindices, and two equality constraints.

$$
\begin{array}{lrcl}
\text{minimize} & \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, X_1 \right\rangle + x_1 & & \\
\text{subject to} & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 & = & 1.0\,, \\
& \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, X_1 \right\rangle + x_0 + x_2 & = & 0.5\,, \\
& x_1 \geq \sqrt{x_0^2 + x_2^2}\,, & & \\
& X_1 \succeq \mathbf{0}\,. & &
\end{array}
\tag{12.7}
$$

The equality constraints are easily rewritten to the conic form, $(g_0, g_1) \in \{0\}^2$, by moving constants such that the right-hand-side becomes zero. The quadratic cone does not fit under the `VAR` keyword in this variable permutation. Instead, it takes a scalar constraint $(g_2, g_3, g_4) = (x_1, x_0, x_2) \in \mathcal{Q}^3$, with scalar variables constructed as $(x_0, x_1, x_2) \in \mathbb{R}^3$. Its formulation in the CBF format is reported in the following list

```
# File written using this version of the Conic Benchmark Format:
#     | Version 1.
VER
1

# The sense of the objective is:
#     | Minimize.
OBJSENSE
MIN

# One PSD variable of this size:
#     | Three times three.
PSDVAR
1
3

# Three scalar variables in this one conic domain:
#     | Three are free.
VAR
3 1
F 3

# Five scalar constraints with affine expressions in two conic domains:
#     | Two are fixed to zero.
#     | Three are in conic quadratic domain.
CON
5 2
L= 2
Q 3

# Five coordinates in F^{obj}_j coefficients:
#     | F^{obj}[0][0,0] = 2.0
#     | F^{obj}[0][1,0] = 1.0
#     | and more...
OBJFCOORD
5
0 0 0 2.0
0 1 0 1.0
0 1 1 2.0
0 2 1 1.0
0 2 2 2.0

# One coordinate in a^{obj}_j coefficients:
#     | a^{obj}[1] = 1.0
OBJACOORD
1
1 1.0

# Nine coordinates in F_ij coefficients:
#     | F[0,0][0,0] = 1.0
#     | F[0,0][1,1] = 1.0
#     | and more...
FCOORD
9
0 0 0 0 1.0
0 0 1 1 1.0
0 0 2 2 1.0
1 0 0 0 1.0
1 0 1 0 1.0
1 0 2 0 1.0
```

```
1 0 1 1 1.0
1 0 2 1 1.0
1 0 2 2 1.0

# Six coordinates in a_ij coefficients:
#     | a[0,1] = 1.0
#     | a[1,0] = 1.0
#     | and more...
ACOORD
6
0 1 1.0
1 0 1.0
1 2 1.0
2 1 1.0
3 0 1.0
4 2 1.0

# Two coordinates in b_i coefficients:
#     | b[0] = -1.0
#     | b[1] = -0.5
BCOORD
2
0 -1.0
1 -0.5
```

Mixing Semidefinite Variables and Linear Matrix Inequalities

The standard forms in semidefinite optimization are usually based either on semidefinite variables or linear matrix inequalities. In the CBF format, both forms are supported and can even be mixed as shown in.

$$\begin{array}{lllll} \text{minimize} & \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 + x_2 + 1 & & \\ \text{subject to} & \left\langle \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X_1 \right\rangle - x_1 - x_2 & \geq & 0.0\,, \\ & x_1 \begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} & \succeq & \mathbf{0}\,, \\ & X_1 \succeq \mathbf{0}\,. & & \end{array} \tag{12.8}$$

Its formulation in the CBF format is written in what follows

```
# File written using this version of the Conic Benchmark Format:
#     | Version 1.
VER
1

# The sense of the objective is:
#     | Minimize.
OBJSENSE
MIN

# One PSD variable of this size:
#     | Two times two.
PSDVAR
1
2

# Two scalar variables in this one conic domain:
#     | Two are free.
VAR
2 1
```

```
F 2

# One PSD constraint of this size:
#     | Two times two.
PSDCON
1
2

# One scalar constraint with an affine expression in this one conic domain:
#     | One is greater than or equal to zero.
CON
1 1
L+ 1

# Two coordinates in F^{obj}_j coefficients:
#     | F^{obj}[0][0,0] = 1.0
#     | F^{obj}[0][1,1] = 1.0
OBJFCOORD
2
0 0 0 1.0
0 1 1 1.0

# Two coordinates in a^{obj}_j coefficients:
#     | a^{obj}[0] = 1.0
#     | a^{obj}[1] = 1.0
OBJACOORD
2
0 1.0
1 1.0

# One coordinate in b^{obj} coefficient:
#     | b^{obj} = 1.0
OBJBCOORD
1.0

# One coordinate in F_ij coefficients:
#     | F[0,0][1,0] = 1.0
FCOORD
1
0 0 1 0 1.0

# Two coordinates in a_ij coefficients:
#     | a[0,0] = -1.0
#     | a[0,1] = -1.0
ACOORD
2
0 0 -1.0
0 1 -1.0

# Four coordinates in H_ij coefficients:
#     | H[0,0][1,0] = 1.0
#     | H[0,0][1,1] = 3.0
#     | and more...
HCOORD
4
0 0 1 0 1.0
0 0 1 1 3.0
0 1 0 0 3.0
0 1 1 0 1.0

# Two coordinates in D_i coefficients:
#     | D[0][0,0] = -1.0
#     | D[0][1,1] = -1.0
```

```
DCOORD
2
0 0 0 -1.0
0 1 1 -1.0
```

Optimization Over a Sequence of Objectives

The linear optimization problem (12.9), is defined for a sequence of objectives such that hotstarting from one to the next might be advantages.

$$\begin{array}{lccl} \text{maximize}_k & g^{obj}_k & & \\ \text{subject to} & 50\,x_0 + 31 & \leq & 250\,, \\ & 3\,x_0 - 2x_1 & \geq & -4\,, \\ & x \in \mathbb{R}^2_+, & & \end{array} \tag{12.9}$$

given,

1. $g^{obj}_0 = x_0 + 0.64x_1$.
2. $g^{obj}_1 = 1.11x_0 + 0.76x_1$.
3. $g^{obj}_2 = 1.11x_0 + 0.85x_1$.

Its formulation in the CBF format is reported in Listing 12.5.

Listing 12.5: Problem (12.9) in CBF format.

```
# File written using this version of the Conic Benchmark Format:
#     | Version 1.
VER
1

# The sense of the objective is:
#     | Maximize.
OBJSENSE
MAX

# Two scalar variables in this one conic domain:
#     | Two are nonnegative.
VAR
2 1
L+ 2

# Two scalar constraints with affine expressions in these two conic domains:
#     | One is in the nonpositive domain.
#     | One is in the nonnegative domain.
CON
2 2
L- 1
L+ 1

# Two coordinates in a^{obj}_j coefficients:
#     | a^{obj}[0] = 1.0
#     | a^{obj}[1] = 0.64
OBJACOORD
2
0 1.0
1 0.64

# Four coordinates in a_ij coefficients:
#     | a[0,0] = 50.0
#     | a[1,0] = 3.0
```

```
#     | and more...
ACOORD
4
0 0 50.0
1 0 3.0
0 1 31.0
1 1 -2.0

# Two coordinates in b_i coefficients:
#     | b[0] = -250.0
#     | b[1] = 4.0
BCOORD
2
0 -250.0
1 4.0

# New problem instance defined in terms of changes.
CHANGE

# Two coordinate changes in a^{obj}_j coefficients. Now it is:
#     | a^{obj}[0] = 1.11
#     | a^{obj}[1] = 0.76
OBJACOORD
2
0 1.11
1 0.76

# New problem instance defined in terms of changes.
CHANGE

# One coordinate change in a^{obj}_j coefficients. Now it is:
#     | a^{obj}[0] = 1.11
#     | a^{obj}[1] = 0.85
OBJACOORD
1
1 0.85
```

12.5 The XML (OSiL) Format

MOSEK can write data in the standard OSiL xml format. For a definition of the OSiL format please see http://www.optimizationservices.org/.

Only linear constraints (possibly with integer variables) are supported. By default output files with the extension `.xml` are written in the OSiL format.

The parameter *MSK_IPAR_WRITE_XML_MODE* controls if the linear coefficients in the A matrix are written in row or column order.

12.6 The Task Format

The Task format is **MOSEK**'s native binary format. It contains a complete image of a **MOSEK** task, i.e.

- Problem data: Linear, conic quadratic, semidefinite and quadratic data
- Problem item names: Variable names, constraints names, cone names etc.
- Parameter settings
- Solutions

There are a few things to be aware of:

- The task format *does not* support General Convex problems since these are defined by arbitrary user-defined functions.
- Status of a solution read from a file will *always* be unknown.
- Parameter settings in a task file *always override* any parameters set on the command line or in a parameter file.

The format is based on the *TAR* (USTar) file format. This means that the individual pieces of data in a `.task` file can be examined by unpacking it as a *TAR* file. Please note that the inverse may not work: Creating a file using *TAR* will most probably not create a valid **MOSEK** Task file since the order of the entries is important.

12.7 The JSON Format

MOSEK provides the possibility to read/write problems in valid JSON format.

> JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

The official JSON website http://www.json.org provides plenty of information along with the format definition.

MOSEK defines two JSON-like formats:

- *jtask*
- *jsol*

> **Warning:** Despite being text-based human-readable formats, *jtask* and *jsol* files will include no indentation and no new-lines, in order to keep the files as compact as possible. We therefore strongly advise to use JSON viewer tools to inspect *jtask* and *jsol* files.

12.7.1 *jtask* format

It stores a problem instance. The *jtask* format contains the same information as a *task format*.

Even though a *jtask* file is human-readable, we do not recommend users to create it by hand, but to rely on **MOSEK**.

12.7.2 *jsol* format

It stores a problem solution. The *jsol* format contains all solutions and information items.

12.7.3 A *jtask* example

In Listing 12.6 we present a file in the *jtask* format that corresponds to the sample problem from `lo1.lp`. The listing has been formatted for readability.

Listing 12.6: A formatted *jtask* file for the `lo1.lp` example.

```
{
    "$schema":"http://mosek.com/json/schema#",
    "Task/INFO":{
        "taskname":"lo1",
        "numvar":4,
        "numcon":3,
        "numcone":0,
        "numbarvar":0,
        "numanz":9,
        "numsymmat":0,
        "mosekver":[
            8,
            0,
            0,
            9
        ]
    },
    "Task/data":{
        "var":{
            "name":[
                "x1",
                "x2",
                "x3",
                "x4"
            ],
            "bk":[
                "lo",
                "ra",
                "lo",
                "lo"
            ],
            "bl":[
                0.0,
                0.0,
                0.0,
                0.0
            ],
            "bu":[
                1e+30,
                1e+1,
                1e+30,
                1e+30
            ],
            "type":[
                "cont",
                "cont",
                "cont",
                "cont"
            ]
        },
        "con":{
            "name":[
                "c1",
                "c2",
                "c3"
            ],
            "bk":[
                "fx",
                "lo",
                "up"
```

```
            ],
            "bl":[
                3e+1,
                1.5e+1,
                    -1e+30
            ],
            "bu":[
                3e+1,
                1e+30,
                2.5e+1
            ]
        },
        "objective":{
            "sense":"max",
            "name":"obj",
            "c":{
                "subj":[
                    0,
                    1,
                    2,
                    3
                ],
                "val":[
                    3e+0,
                    1e+0,
                    5e+0,
                    1e+0
                ]
            },
            "cfix":0.0
        },
        "A":{
            "subi":[
                0,
                0,
                0,
                1,
                1,
                1,
                1,
                2,
                2
            ],
            "subj":[
                0,
                1,
                2,
                0,
                1,
                2,
                3,
                1,
                3
            ],
            "val":[
                3e+0,
                1e+0,
                2e+0,
                2e+0,
                1e+0,
                3e+0,
                1e+0,
                2e+0,
```

```
                3e+0
            ]
        }
    },
    "Task/parameters":{
        "iparam":{
            "ANA_SOL_BASIS":"ON",
            "ANA_SOL_PRINT_VIOLATED":"OFF",
            "AUTO_SORT_A_BEFORE_OPT":"OFF",
            "AUTO_UPDATE_SOL_INFO":"OFF",
            "BASIS_SOLVE_USE_PLUS_ONE":"OFF",
            "BI_CLEAN_OPTIMIZER":"OPTIMIZER_FREE",
            "BI_IGNORE_MAX_ITER":"OFF",
            "BI_IGNORE_NUM_ERROR":"OFF",
            "BI_MAX_ITERATIONS":1000000,
            "CACHE_LICENSE":"ON",
            "CHECK_CONVEXITY":"CHECK_CONVEXITY_FULL",
            "COMPRESS_STATFILE":"ON",
            "CONCURRENT_NUM_OPTIMIZERS":2,
            "CONCURRENT_PRIORITY_DUAL_SIMPLEX":2,
            "CONCURRENT_PRIORITY_FREE_SIMPLEX":3,
            "CONCURRENT_PRIORITY_INTPNT":4,
            "CONCURRENT_PRIORITY_PRIMAL_SIMPLEX":1,
            "FEASREPAIR_OPTIMIZE":"FEASREPAIR_OPTIMIZE_NONE",
            "INFEAS_GENERIC_NAMES":"OFF",
            "INFEAS_PREFER_PRIMAL":"ON",
            "INFEAS_REPORT_AUTO":"OFF",
            "INFEAS_REPORT_LEVEL":1,
            "INTPNT_BASIS":"BI_ALWAYS",
            "INTPNT_DIFF_STEP":"ON",
            "INTPNT_FACTOR_DEBUG_LVL":0,
            "INTPNT_FACTOR_METHOD":0,
            "INTPNT_HOTSTART":"INTPNT_HOTSTART_NONE",
            "INTPNT_MAX_ITERATIONS":400,
            "INTPNT_MAX_NUM_COR":-1,
            "INTPNT_MAX_NUM_REFINEMENT_STEPS":-1,
            "INTPNT_OFF_COL_TRH":40,
            "INTPNT_ORDER_METHOD":"ORDER_METHOD_FREE",
            "INTPNT_REGULARIZATION_USE":"ON",
            "INTPNT_SCALING":"SCALING_FREE",
            "INTPNT_SOLVE_FORM":"SOLVE_FREE",
            "INTPNT_STARTING_POINT":"STARTING_POINT_FREE",
            "LIC_TRH_EXPIRY_WRN":7,
            "LICENSE_DEBUG":"OFF",
            "LICENSE_PAUSE_TIME":0,
            "LICENSE_SUPPRESS_EXPIRE_WRNS":"OFF",
            "LICENSE_WAIT":"OFF",
            "LOG":10,
            "LOG_ANA_PRO":1,
            "LOG_BI":4,
            "LOG_BI_FREQ":2500,
            "LOG_CHECK_CONVEXITY":0,
            "LOG_CONCURRENT":1,
            "LOG_CUT_SECOND_OPT":1,
            "LOG_EXPAND":0,
            "LOG_FACTOR":1,
            "LOG_FEAS_REPAIR":1,
            "LOG_FILE":1,
            "LOG_HEAD":1,
            "LOG_INFEAS_ANA":1,
            "LOG_INTPNT":4,
            "LOG_MIO":4,
            "LOG_MIO_FREQ":1000,
```

```
            "LOG_OPTIMIZER":1,
            "LOG_ORDER":1,
            "LOG_PRESOLVE":1,
            "LOG_RESPONSE":0,
            "LOG_SENSITIVITY":1,
            "LOG_SENSITIVITY_OPT":0,
            "LOG_SIM":4,
            "LOG_SIM_FREQ":1000,
            "LOG_SIM_MINOR":1,
            "LOG_STORAGE":1,
            "MAX_NUM_WARNINGS":10,
            "MIO_BRANCH_DIR":"BRANCH_DIR_FREE",
            "MIO_CONSTRUCT_SOL":"OFF",
            "MIO_CUT_CLIQUE":"ON",
            "MIO_CUT_CMIR":"ON",
            "MIO_CUT_GMI":"ON",
            "MIO_CUT_KNAPSACK_COVER":"OFF",
            "MIO_HEURISTIC_LEVEL":-1,
            "MIO_MAX_NUM_BRANCHES":-1,
            "MIO_MAX_NUM_RELAXS":-1,
            "MIO_MAX_NUM_SOLUTIONS":-1,
            "MIO_MODE":"MIO_MODE_SATISFIED",
            "MIO_MT_USER_CB":"ON",
            "MIO_NODE_OPTIMIZER":"OPTIMIZER_FREE",
            "MIO_NODE_SELECTION":"MIO_NODE_SELECTION_FREE",
            "MIO_PERSPECTIVE_REFORMULATE":"ON",
            "MIO_PROBING_LEVEL":-1,
            "MIO_RINS_MAX_NODES":-1,
            "MIO_ROOT_OPTIMIZER":"OPTIMIZER_FREE",
            "MIO_ROOT_REPEAT_PRESOLVE_LEVEL":-1,
            "MT_SPINCOUNT":0,
            "NUM_THREADS":0,
            "OPF_MAX_TERMS_PER_LINE":5,
            "OPF_WRITE_HEADER":"ON",
            "OPF_WRITE_HINTS":"ON",
            "OPF_WRITE_PARAMETERS":"OFF",
            "OPF_WRITE_PROBLEM":"ON",
            "OPF_WRITE_SOL_BAS":"ON",
            "OPF_WRITE_SOL_ITG":"ON",
            "OPF_WRITE_SOL_ITR":"ON",
            "OPF_WRITE_SOLUTIONS":"OFF",
            "OPTIMIZER":"OPTIMIZER_FREE",
            "PARAM_READ_CASE_NAME":"ON",
            "PARAM_READ_IGN_ERROR":"OFF",
            "PRESOLVE_ELIMINATOR_MAX_FILL":-1,
            "PRESOLVE_ELIMINATOR_MAX_NUM_TRIES":-1,
            "PRESOLVE_LEVEL":-1,
            "PRESOLVE_LINDEP_ABS_WORK_TRH":100,
            "PRESOLVE_LINDEP_REL_WORK_TRH":100,
            "PRESOLVE_LINDEP_USE":"ON",
            "PRESOLVE_MAX_NUM_REDUCTIONS":-1,
            "PRESOLVE_USE":"PRESOLVE_MODE_FREE",
            "PRIMAL_REPAIR_OPTIMIZER":"OPTIMIZER_FREE",
            "QO_SEPARABLE_REFORMULATION":"OFF",
            "READ_DATA_COMPRESSED":"COMPRESS_FREE",
            "READ_DATA_FORMAT":"DATA_FORMAT_EXTENSION",
            "READ_DEBUG":"OFF",
            "READ_KEEP_FREE_CON":"OFF",
            "READ_LP_DROP_NEW_VARS_IN_BOU":"OFF",
            "READ_LP_QUOTED_NAMES":"ON",
            "READ_MPS_FORMAT":"MPS_FORMAT_FREE",
            "READ_MPS_WIDTH":1024,
            "READ_TASK_IGNORE_PARAM":"OFF",
```

```
            "SENSITIVITY_ALL":"OFF",
            "SENSITIVITY_OPTIMIZER":"OPTIMIZER_FREE_SIMPLEX",
            "SENSITIVITY_TYPE":"SENSITIVITY_TYPE_BASIS",
            "SIM_BASIS_FACTOR_USE":"ON",
            "SIM_DEGEN":"SIM_DEGEN_FREE",
            "SIM_DUAL_CRASH":90,
            "SIM_DUAL_PHASEONE_METHOD":0,
            "SIM_DUAL_RESTRICT_SELECTION":50,
            "SIM_DUAL_SELECTION":"SIM_SELECTION_FREE",
            "SIM_EXPLOIT_DUPVEC":"SIM_EXPLOIT_DUPVEC_OFF",
            "SIM_HOTSTART":"SIM_HOTSTART_FREE",
            "SIM_HOTSTART_LU":"ON",
            "SIM_INTEGER":0,
            "SIM_MAX_ITERATIONS":10000000,
            "SIM_MAX_NUM_SETBACKS":250,
            "SIM_NON_SINGULAR":"ON",
            "SIM_PRIMAL_CRASH":90,
            "SIM_PRIMAL_PHASEONE_METHOD":0,
            "SIM_PRIMAL_RESTRICT_SELECTION":50,
            "SIM_PRIMAL_SELECTION":"SIM_SELECTION_FREE",
            "SIM_REFACTOR_FREQ":0,
            "SIM_REFORMULATION":"SIM_REFORMULATION_OFF",
            "SIM_SAVE_LU":"OFF",
            "SIM_SCALING":"SCALING_FREE",
            "SIM_SCALING_METHOD":"SCALING_METHOD_POW2",
            "SIM_SOLVE_FORM":"SOLVE_FREE",
            "SIM_STABILITY_PRIORITY":50,
            "SIM_SWITCH_OPTIMIZER":"OFF",
            "SOL_FILTER_KEEP_BASIC":"OFF",
            "SOL_FILTER_KEEP_RANGED":"OFF",
            "SOL_READ_NAME_WIDTH":-1,
            "SOL_READ_WIDTH":1024,
            "SOLUTION_CALLBACK":"OFF",
            "TIMING_LEVEL":1,
            "WRITE_BAS_CONSTRAINTS":"ON",
            "WRITE_BAS_HEAD":"ON",
            "WRITE_BAS_VARIABLES":"ON",
            "WRITE_DATA_COMPRESSED":0,
            "WRITE_DATA_FORMAT":"DATA_FORMAT_EXTENSION",
            "WRITE_DATA_PARAM":"OFF",
            "WRITE_FREE_CON":"OFF",
            "WRITE_GENERIC_NAMES":"OFF",
            "WRITE_GENERIC_NAMES_IO":1,
            "WRITE_IGNORE_INCOMPATIBLE_CONIC_ITEMS":"OFF",
            "WRITE_IGNORE_INCOMPATIBLE_ITEMS":"OFF",
            "WRITE_IGNORE_INCOMPATIBLE_NL_ITEMS":"OFF",
            "WRITE_IGNORE_INCOMPATIBLE_PSD_ITEMS":"OFF",
            "WRITE_INT_CONSTRAINTS":"ON",
            "WRITE_INT_HEAD":"ON",
            "WRITE_INT_VARIABLES":"ON",
            "WRITE_LP_FULL_OBJ":"ON",
            "WRITE_LP_LINE_WIDTH":80,
            "WRITE_LP_QUOTED_NAMES":"ON",
            "WRITE_LP_STRICT_FORMAT":"OFF",
            "WRITE_LP_TERMS_PER_LINE":10,
            "WRITE_MPS_FORMAT":"MPS_FORMAT_FREE",
            "WRITE_MPS_INT":"ON",
            "WRITE_PRECISION":15,
            "WRITE_SOL_BARVARIABLES":"ON",
            "WRITE_SOL_CONSTRAINTS":"ON",
            "WRITE_SOL_HEAD":"ON",
            "WRITE_SOL_IGNORE_INVALID_NAMES":"OFF",
            "WRITE_SOL_VARIABLES":"ON",
```

```
            "WRITE_TASK_INC_SOL":"ON",
            "WRITE_XML_MODE":"WRITE_XML_MODE_ROW"
        },
        "dparam":{
            "ANA_SOL_INFEAS_TOL":1e-6,
            "BASIS_REL_TOL_S":1e-12,
            "BASIS_TOL_S":1e-6,
            "BASIS_TOL_X":1e-6,
            "CHECK_CONVEXITY_REL_TOL":1e-10,
            "DATA_TOL_AIJ":1e-12,
            "DATA_TOL_AIJ_HUGE":1e+20,
            "DATA_TOL_AIJ_LARGE":1e+10,
            "DATA_TOL_BOUND_INF":1e+16,
            "DATA_TOL_BOUND_WRN":1e+8,
            "DATA_TOL_C_HUGE":1e+16,
            "DATA_TOL_CJ_LARGE":1e+8,
            "DATA_TOL_QIJ":1e-16,
            "DATA_TOL_X":1e-8,
            "FEASREPAIR_TOL":1e-10,
            "INTPNT_CO_TOL_DFEAS":1e-8,
            "INTPNT_CO_TOL_INFEAS":1e-10,
            "INTPNT_CO_TOL_MU_RED":1e-8,
            "INTPNT_CO_TOL_NEAR_REL":1e+3,
            "INTPNT_CO_TOL_PFEAS":1e-8,
            "INTPNT_CO_TOL_REL_GAP":1e-7,
            "INTPNT_NL_MERIT_BAL":1e-4,
            "INTPNT_NL_TOL_DFEAS":1e-8,
            "INTPNT_NL_TOL_MU_RED":1e-12,
            "INTPNT_NL_TOL_NEAR_REL":1e+3,
            "INTPNT_NL_TOL_PFEAS":1e-8,
            "INTPNT_NL_TOL_REL_GAP":1e-6,
            "INTPNT_NL_TOL_REL_STEP":9.95e-1,
            "INTPNT_QO_TOL_DFEAS":1e-8,
            "INTPNT_QO_TOL_INFEAS":1e-10,
            "INTPNT_QO_TOL_MU_RED":1e-8,
            "INTPNT_QO_TOL_NEAR_REL":1e+3,
            "INTPNT_QO_TOL_PFEAS":1e-8,
            "INTPNT_QO_TOL_REL_GAP":1e-8,
            "INTPNT_TOL_DFEAS":1e-8,
            "INTPNT_TOL_DSAFE":1e+0,
            "INTPNT_TOL_INFEAS":1e-10,
            "INTPNT_TOL_MU_RED":1e-16,
            "INTPNT_TOL_PATH":1e-8,
            "INTPNT_TOL_PFEAS":1e-8,
            "INTPNT_TOL_PSAFE":1e+0,
            "INTPNT_TOL_REL_GAP":1e-8,
            "INTPNT_TOL_REL_STEP":9.999e-1,
            "INTPNT_TOL_STEP_SIZE":1e-6,
            "LOWER_OBJ_CUT":-1e+30,
            "LOWER_OBJ_CUT_FINITE_TRH":-5e+29,
            "MIO_DISABLE_TERM_TIME":-1e+0,
            "MIO_MAX_TIME":-1e+0,
            "MIO_MAX_TIME_APRX_OPT":6e+1,
            "MIO_NEAR_TOL_ABS_GAP":0.0,
            "MIO_NEAR_TOL_REL_GAP":1e-3,
            "MIO_REL_GAP_CONST":1e-10,
            "MIO_TOL_ABS_GAP":0.0,
            "MIO_TOL_ABS_RELAX_INT":1e-5,
            "MIO_TOL_FEAS":1e-6,
            "MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT":0.0,
            "MIO_TOL_REL_GAP":1e-4,
            "MIO_TOL_X":1e-6,
            "OPTIMIZER_MAX_TIME":-1e+0,
```

```
            "PRESOLVE_TOL_ABS_LINDEP":1e-6,
            "PRESOLVE_TOL_AIJ":1e-12,
            "PRESOLVE_TOL_REL_LINDEP":1e-10,
            "PRESOLVE_TOL_S":1e-8,
            "PRESOLVE_TOL_X":1e-8,
            "QCQO_REFORMULATE_REL_DROP_TOL":1e-15,
            "SEMIDEFINITE_TOL_APPROX":1e-10,
            "SIM_LU_TOL_REL_PIV":1e-2,
            "SIMPLEX_ABS_TOL_PIV":1e-7,
            "UPPER_OBJ_CUT":1e+30,
            "UPPER_OBJ_CUT_FINITE_TRH":5e+29
        },
        "sparam":{
            "BAS_SOL_FILE_NAME":"",
            "DATA_FILE_NAME":"examples/tools/data/lo1.mps",
            "DEBUG_FILE_NAME":"",
            "INT_SOL_FILE_NAME":"",
            "ITR_SOL_FILE_NAME":"",
            "MIO_DEBUG_STRING":"",
            "PARAM_COMMENT_SIGN":"%%",
            "PARAM_READ_FILE_NAME":"",
            "PARAM_WRITE_FILE_NAME":"",
            "READ_MPS_BOU_NAME":"",
            "READ_MPS_OBJ_NAME":"",
            "READ_MPS_RAN_NAME":"",
            "READ_MPS_RHS_NAME":"",
            "SENSITIVITY_FILE_NAME":"",
            "SENSITIVITY_RES_FILE_NAME":"",
            "SOL_FILTER_XC_LOW":"",
            "SOL_FILTER_XC_UPR":"",
            "SOL_FILTER_XX_LOW":"",
            "SOL_FILTER_XX_UPR":"",
            "STAT_FILE_NAME":"",
            "STAT_KEY":"",
            "STAT_NAME":"",
            "WRITE_LP_GEN_VAR_NAME":"XMSKGEN"
        }
    }
}
```

12.8 The Solution File Format

MOSEK provides several solution files depending on the problem type and the optimizer used:

- *basis solution file* (extension `.bas`) if the problem is optimized using the simplex optimizer or basis identification is performed,
- *interior solution file* (extension `.sol`) if a problem is optimized using the interior-point optimizer and no basis identification is required,
- *integer solution file* (extension `.int`) if the problem contains integer constrained variables.

All solution files have the format:

```
NAME                : <problem name>
PROBLEM STATUS      : <status of the problem>
SOLUTION STATUS     : <status of the solution>
OBJECTIVE NAME      : <name of the objective function>
PRIMAL OBJECTIVE    : <primal objective value corresponding to the solution>
DUAL OBJECTIVE      : <dual objective value corresponding to the solution>
CONSTRAINTS
```

```
INDEX   NAME     AT ACTIVITY    LOWER LIMIT   UPPER LIMIT   DUAL LOWER   DUAL UPPER
?       <name>   ?? <a value>   <a value>     <a value>     <a value>    <a value>
VARIABLES
INDEX   NAME     AT ACTIVITY    LOWER LIMIT   UPPER LIMIT   DUAL LOWER   DUAL UPPER    CONIC␣
↪DUAL
?       <name>   ?? <a value>   <a value>     <a value>     <a value>    <a value>     <a value>
```

In the example the fields `?` and `<>` will be filled with problem and solution specific information. As can be observed a solution report consists of three sections, i.e.

- `HEADER` In this section, first the name of the problem is listed and afterwards the problem and solution status are shown. Next the primal and dual objective values are displayed.
- `CONSTRAINTS` For each constraint i of the form

$$l_i^c \leq \sum_{j=1}^{n} a_{ij} x_j \leq u_i^c, \tag{12.10}$$

  the following information is listed:

  – `INDEX`: A sequential index assigned to the constraint by **MOSEK**

  – `NAME`: The name of the constraint assigned by the user.

  – `AT`: The status of the constraint. In Table 12.4 the possible values of the status keys and their interpretation are shown.

Table 12.4: Status keys.

Status key	Interpretation
`UN`	Unknown status
`BS`	Is basic
`SB`	Is superbasic
`LL`	Is at the lower limit (bound)
`UL`	Is at the upper limit (bound)
`EQ`	Lower limit is identical to upper limit
`**`	Is infeasible i.e. the lower limit is greater than the upper limit.

  – `ACTIVITY`: the quantity $\sum_{j=1}^{n} a_{ij} x_j^*$, where x^* is the value of the primal solution.

  – `LOWER LIMIT`: the quantity l_i^c (see (12.10).)

  – `UPPER LIMIT`: the quantity u_i^c (see (12.10).)

  – `DUAL LOWER`: the dual multiplier corresponding to the lower limit on the constraint.

  – `DUAL UPPER`: the dual multiplier corresponding to the upper limit on the constraint.

- `VARIABLES` The last section of the solution report lists information about the variables. This information has a similar interpretation as for the constraints. However, the column with the header `CONIC DUAL` is included for problems having one or more conic constraints. This column shows the dual variables corresponding to the conic constraints.

Example: `lo1.sol`

In Listing 12.7 we show the solution file for the `lo1.opf` problem.

Listing 12.7: An example of `.sol` file.

```
NAME                :
PROBLEM STATUS      : PRIMAL_AND_DUAL_FEASIBLE
SOLUTION STATUS     : OPTIMAL
OBJECTIVE NAME      : obj
```

```
PRIMAL OBJECTIVE     : 8.33333333e+01
DUAL OBJECTIVE       : 8.33333332e+01

CONSTRAINTS
INDEX      NAME           AT ACTIVITY                LOWER LIMIT        UPPER LIMIT        ␣
↪DUAL LOWER                DUAL UPPER
0          c1             EQ 3.00000000000000e+01    3.00000000e+01     3.00000000e+01     -0.
↪00000000000000e+00    -2.49999999741654e+00
1          c2             SB 5.33333333049188e+01    1.50000000e+01     NONE               2.
↪09157603759397e-10     -0.00000000000000e+00
2          c3             UL 2.49999999842049e+01    NONE               2.50000000e+01     -0.
↪00000000000000e+00    -3.33333332895110e-01

VARIABLES
INDEX      NAME           AT ACTIVITY                LOWER LIMIT        UPPER LIMIT        ␣
↪DUAL LOWER                DUAL UPPER
0          x1             LL 1.67020427073508e-09    0.00000000e+00     NONE               -4.
↪49999999528055e+00    -0.00000000000000e+00
1          x2             LL 2.93510446280504e-09    0.00000000e+00     1.00000000e+01     -2.
↪16666666494916e+00    6.20863861687316e-10
2          x3             SB 1.49999999899425e+01    0.00000000e+00     NONE               -8.
↪79123177454657e-10    -0.00000000000000e+00
3          x4             SB 8.33333332273116e+00    0.00000000e+00     NONE               -1.
↪69795978899185e-09    -0.00000000000000e+00
```

BIBLIOGRAPHY

[Naz87] J. L. Nazareth. *Computer Solution of Linear Programs*. Oxford University Press, New York, 1987.

SYMBOL INDEX

Functions

Parameters

Response codes

INDEX

Symbols

A

B

C

E

F