



MOSEK Optimization Server

Release 9.3.12

MOSEK ApS

14 January 2022

Contents

1	Introduction	1
1.1	Why the Optimization Server?	2
2	Contact Information	3
3	License Agreement	4
4	Installation	8
4.1	Try it out	8
4.2	Requirements	8
4.3	Locating files	8
4.4	Installation	9
5	Overview	11
5.1	Synchronous Optimization	11
5.2	Asynchronous Optimization	11
5.3	With or without the MOSEK API	13
5.4	Open or encrypted mode	13
6	Guidelines	14
6.1	Technical guidelines	14
6.2	The license system	15
6.3	Security	15
7	REST API tutorials	16
7.1	Synchronous Problem Submission	16
7.2	Asynchronous Problem Submission	21
8	Web GUI interface	26
8.1	Setup	26
8.2	Users	26
8.3	Tokens	26
8.4	Jobs	28
8.5	User's profile	28
9	OptServer Reference	29
9.1	OptServer REST API	29
9.2	Parameters grouped by topic	31
9.3	Parameters (alphabetical list sorted by type)	43
9.4	Response codes	82
9.5	Constants	100
10	Supported File Formats	124
10.1	The LP File Format	125
10.2	The MPS File Format	130
10.3	The OPF Format	142
10.4	The CBF Format	152
10.5	The PTF Format	166

10.6	The Task Format	171
10.7	The JSON Format	171
10.8	The Solution File Format	179
Bibliography		182
Symbol Index		183
Index		190

Chapter 1

Introduction

The **MOSEK** Optimization Suite 9.3.12 is a powerful software package capable of solving large-scale optimization problems of the following kind:

- linear,
- conic:
 - conic quadratic (also known as second-order cone),
 - involving the exponential cone,
 - involving the power cone,
 - semidefinite,
- convex quadratic and quadratically constrained,
- integer.

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

$$Ax - b \geq 0.$$

In conic optimization this is replaced with a wider class of constraints

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

where \mathcal{K} is a *convex cone*. For example in 3 dimensions \mathcal{K} may correspond to an ice cream cone. The conic optimizer in **MOSEK** supports a number of different types of cones \mathcal{K} , which allows a surprisingly large number of nonlinear relations to be modeled, 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 receives optimization tasks, solves them, and returns solution and log information. A typical application would be offloading heavy computations from client computers, when the problem is set up, to a remote powerful machine, and returning solutions back.

The OptServer can be used in a few ways:

- Users of the Optimizer and Fusion API can use the OptServer directly from the API by providing the server, port number and credentials (if appropriate). This way then can switch between running the same optimization locally and remotely with no change to the rest of their **MOSEK** code except for the `optimize` or `solve` call.
- Similarly to the above, but in asynchronous mode, where the local call does not wait for the remote optimization to terminate. Instead the user should periodically poll the server for a solution.
- Optimization models in standard file formats (MPS, LP, CBF, OPF, **MOSEK** task) can also be sent to the server using a REST API over HTTP or HTTPS and the server returns a file with the solution.

The documentation of the relevant Optimizer API contains examples of calling the remote server using the first two API-based methods.

Chapter 2

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	https://blog.mosek.com/
Google Group	https://groups.google.com/forum/#!forum/mosek
Twitter	https://twitter.com/mosektw
Linkedin	https://www.linkedin.com/company/mosek-aps
Youtube	https://www.youtube.com/channel/UCvIyectEVLp31NXeD5mIbEw

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

Chapter 3

License Agreement

Before using the **MOSEK** software, please read the license agreement available in the distribution at <MSKHOME>/mosek/9.3/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.

```
/*
 *
 * The author of this software is David M. Gay.
 *
 * Copyright (c) 1991, 2000, 2001 by Lucent Technologies.
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose without fee is hereby granted, provided that this entire notice
 * is included in all copies of any software which is or includes a copy
 * or modification of this software and in all copies of the supporting
 * documentation for such software.
 *
 * THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED
 * WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY
 * REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY
 * OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.
 *
 *****/
```

Zstandard

MOSEK includes the *Zstandard* library developed by Facebook obtained from [github/zstd](#). The license agreement for *Zstandard* is shown in [Listing 3.3](#).

Listing 3.3: *Zstandard* license.

```
BSD License

For Zstandard software

Copyright (c) 2016-present, Facebook, Inc. All rights reserved.

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

* Neither the name Facebook nor the names of its contributors may be used to
  endorse or promote products derived from this software without specific
  prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
```

(continues on next page)

(continued from previous page)

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

mimalloc

MOSEK includes the *mimalloc* memory allocator library from [github/mimalloc](https://github.com/mimalloc). The license agreement for *mimalloc* is shown in [Listing 3.4](#).

Listing 3.4: *mimalloc* license.

MIT License

Copyright (c) 2019 Microsoft Corporation, Daan Leijen

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

blis

MOSEK includes the *blis* implementation of BLAS library from [github/blis](https://github.com/blis). The license agreement for *blis* is shown in [Listing 3.5](#).

Listing 3.5: *blis* license.

Copyright (C) 2018, The University of Texas at Austin
Copyright (C) 2016, Hewlett Packard Enterprise Development LP
Copyright (C) 2018 - 2019, Advanced Micro Devices, Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name(s) of the copyright holder(s) nor the names of its contributors may be used to endorse or promote products derived

(continues on next page)

from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

OpenBLAS

MOSEK includes the *OpenBLAS* implementation of BLAS library from [github/OpenBLAS](https://github.com/OpenBLAS). The license agreement for *OpenBLAS* is shown in [Listing 3.6](#).

Listing 3.6: *openblas* license.

Copyright (c) 2011-2014, The OpenBLAS Project
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the OpenBLAS project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Chapter 4

Installation

4.1 Try it out

It is possible to quickly try out the OptServer without going through the full installation process. The two options are:

- Use the public instance running at <https://solve.mosek.com>. It is a demo with size limitations suited for testing small problems. The website has instructions and coordinates of access points.
- Use the Docker image available from <https://github.com/MOSEK/Dockerfiles/tree/master/optserver-demo>. It will install the simplest stand-alone OptServer container with latest **MOSEK** version and with all dependencies installed internally.

The rest of this page describes the complete installation process.

4.2 Requirements

The following are prerequisites to run OptServer:

- OptServer is only available for 64bit Linux.
- Access to a PostgreSQL database is required.
- **MOSEK** binaries are required. If the OptServer is installed from a standard **MOSEK** distribution, then it will naturally contain the necessary files, however an external **MOSEK** installation can also be used.

4.3 Locating files

The relevant files of the Optimization Server are organized as reported below

Table 4.1: Relevant files for the Optimization Server.

Relative Path	Description
<MSKHOME>/mosek/9.3/opt-server/bin	Scripts and binaries
<MSKHOME>/mosek/9.3/opt-server/etc	Configuration files and certificates
<MSKHOME>/mosek/9.3/opt-server/var	Runtime directory, Web interface

where <MSKHOME> is the folder in which the **MOSEK** Optimization Suite has been installed.

4.4 Installation

To install OptServer and test the installation perform the following steps.

4.4.1 Run install script

Run the script `<MSKHOME>/mosek/9.3/opt-server/bin/install_MosekServer` to configure the server. The full list of supported options can be obtained via

```
./install_MosekServer --help
```

As a reasonable minimum the configuration should specify where to install the server, the port to listen on, the database connection string and the location of an available **MOSEK** installation. For example:

```
./install_MosekServer --inplace \  
                      --port $PORT \  
                      --mosekdir ../../.. \  
                      --database-resource "host=/var/run/postgresql user=$USER dbname=  
→$DBNAME sslmode=disable"
```

The install script creates a configuration file `<MSKHOME>/mosek/9.3/opt-server/etc/Mosek/server.conf` which can be edited by hand if necessary.

4.4.2 Initialize the database

Run

```
./MosekServer --create-database
```

to initialize the database. It will use the information provided in the database connection string specified in the previous step. This step is not necessary if the database exists from a previous installation.

4.4.3 Initialize admin password (optional)

If the Web interface to the OptServer is to be used, run

```
./MosekServer --reset-admin
```

to set the password for the **admin** user. This step is not required if the Web GUI will not be used. Note that the GUI will only be available with SSL enabled, see [Sec. 6.3](#).

4.4.4 Run the server

Start the server by running the script

```
./MosekServer
```

The server will print its initial configuration and continue writing the log to a file. To obtain the full list of server options run

```
./MosekServer --help
```

For debugging purposes it is convenient to use the options

```
./MosekServer --debug --logfile -
```

This will increase the amount of debug output and redirect it to standard output.

4.4.5 Test the installation

To test that the OptServer is working properly locate, compile (if necessary) and run the example `opt_server_sync.*` for either C, Python, Java or C#. Examples and sample data files can be found in the distribution package under `<MSKHOME>/mosek/9.3/tools/examples`.

For example, assuming that **MOSEK** was installed in Python, one can go to the folder with Python examples and run

```
python opt_server_sync.py ../data/25fv47.mps $SERVER $PORT
```

where `$SERVER:$PORT` points to the OptServer. If the configuration is correct the example will print a log from solving the problem and a solution summary. In case of issues the log output of the OptServer should be consulted to determine the cause.

This example also demonstrates how to use the OptServer from the **MOSEK** API.

Chapter 5

Overview

In this section we present the basic mechanism of the OptServer.

- Sec. 5.1
- Sec. 5.2
- Sec. 5.3
- Sec. 5.4

5.1 Synchronous Optimization

The easiest way to submit an optimization problem to the OptServer is in *synchronous mode*, where the caller is blocked while waiting for the optimization:

1. A submission request is sent over to 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 result and passes over to the client.
5. The client receives the solution and resumes.

The process can be represented as in Fig. ??.

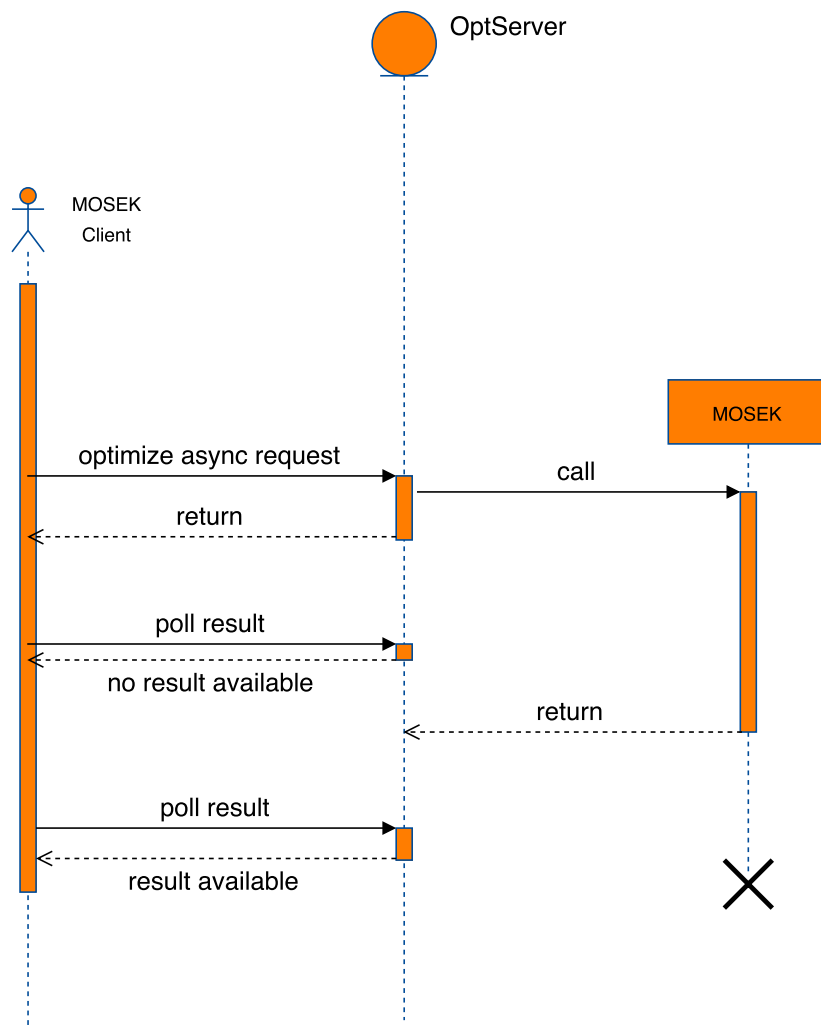
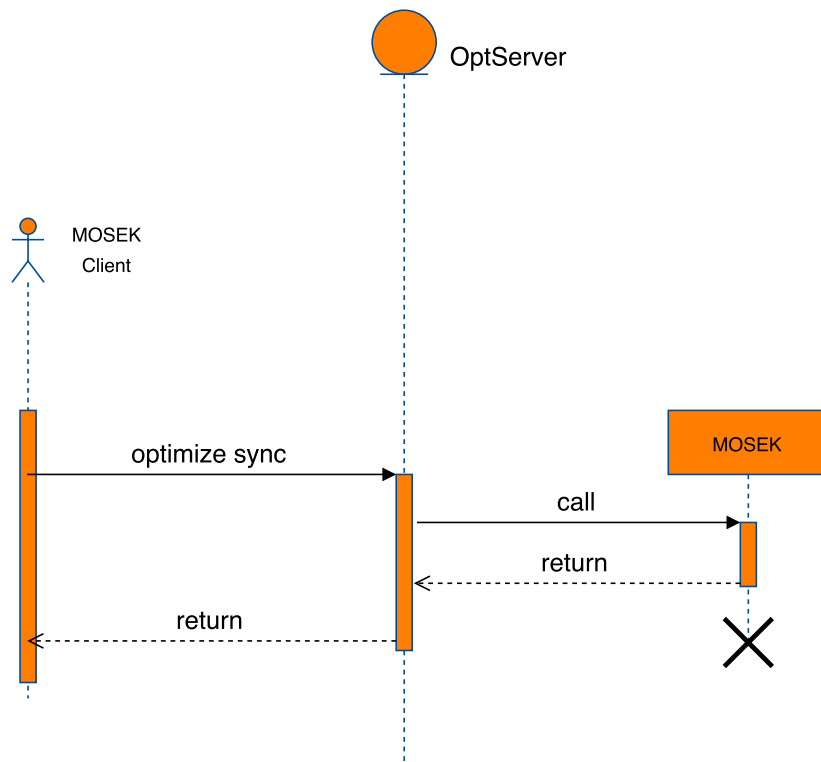
This workflow has the following advantages:

- It is effective for problems where the solution is expected reasonably quickly.
- The changes to the code compared to a local optimization are almost nonexistent. They boil down to invoking a different method in place of the usual `optimize` or similar.

5.2 Asynchronous Optimization

The OptServer accepts jobs also in *asynchronous mode*, where the client is not blocked while waiting for the result:

1. A submission request is sent over to the OptServer and the problem is transferred.
2. The client regains control and continues its own execution flow.
3. The client can poll 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 results, which are available to the client next time it queries.



The process can be represented as in Fig. ??.
Asynchronous mode is particularly suitable when

- A job is expected to run for long time.
- One wants to submit a set of jobs to run in parallel.
- The submitter is a short-lived process.

5.3 With or without the MOSEK API

Calling OptServer using the MOSEK API

The **MOSEK** API provides an interface to invoke the OptServer from the client, both in synchronous and asynchronous mode. It is currently available for the Optimizer API (synchronous and asynchronous) and Fusion (synchronous). The API is a set of functions such as `optimizermt`, `asyncoptimize`, `asyncpoll` and similar, which form a replacement for the standard `optimize` call, while the rest of the **MOSEK** code (creating task, loading data, retrieving results) remains the same. The details and examples can be found in the manuals for the Optimizer and Fusion APIs. It is possible to retrieve the log via a log handler and to interrupt a solver from a callback handler also during remote optimization.

Calling OptServer directly

Alternatively it is possible to call the OptServer through a REST API, submitting an optimization problem in one of formats supported by **MOSEK**. In this case the caller is responsible for assembling the data, communicating with the solver and interpreting the answer. Details and examples can be found in [Sec. 7](#). Using this approach it is possible to perform optimization from environments that cannot support a **MOSEK** client, for example from a Web application.

5.4 Open or encrypted mode

The server can be used in two modes.

Fully open

If no SSL keys are provided at installation, the server will run using HTTP only, without any user authentication, with anonymous job submission open to everybody and without the possibility to use the Web GUI. **This is the only mode supported by the MOSEK client API up to version 9.1.**

Encrypted

If SSL keys are provided at installation, the server will run using HTTPS only (also for job submission). Users need to be authenticated (for example via tokens) unless anonymous submission is explicitly enabled. In this mode it is possible to enable the Web GUI. See [Sec. 6.3](#).

Chapter 6

Guidelines

6.1 Technical guidelines

Modularity

The OptServer is a very lightweight service that handles requests as follows:

- receive a request,
- save the submitted problem to disk,
- run a command-line version of **MOSEK** (`mosekcli`) to solve the problem and save results on disk,
- provide the solution to the caller.

In particular a **MOSEK** installation including the `mosekcli` binary is required to run OptServer. Typically one would use the **MOSEK** binary from the same distribution package from which the OptServer was installed, but the setup is modular and it is possible to use any other **MOSEK** version. In particular updating the solver can be performed independently of updating the OptServer binaries.

Network load

Most of the network load is due to the transfer of the optimization problem from the client to the server. Therefore for long running jobs the transfer time is typically negligible, but for very small problems it will be a significant part of the solution time.

Disk and database

The database is used to store information about jobs (optimization tasks) as well as user information if using the Web GUI. Actual jobs are stored on disk along with log and solutions. Each submitted job is allocated a folder in `var/Mosek/jobs/tasks`. Therefore, in case of a problem, the status and solution can be recovered from disk.

A suitable amount of free space must be available. OptServer does not delete data for completed jobs.

Load balancing

The OptServer does not implement any load balancing. It launches jobs as requests come along. Users should be careful not to overcommit the CPUs and memory, and ensure there is a sufficient number of licenses (see below).

6.2 The license system

MOSEK is a commercial product that **always** needs a valid license to work. **MOSEK** uses a third party license manager to implement license checking. The number of license tokens provided determines the number of optimizations that can be run simultaneously.

A **MOSEK** license must be available on the machine which hosts the OptServer. Each job submitted to the OptServer will be solved by a new solver process, hence it will require a new license checkout. If the license is not unlimited, then the number of tokens determines the maximal number of jobs that can run simultaneously. In this case setting the license wait flag with the parameter *MSK_IPAR_LICENSE_WAIT* will force **MOSEK** to wait until a license token becomes available instead of returning with an error.

6.3 Security

The Web GUI of the OptServer uses HTTPS. To enable the GUI the user must point the installation script to a folder with `cert.pem` and `key.pem` files, for example:

```
./install_MosekServer --inplace \  
                      --port $PORT \  
                      --mosekdir ../../.. \  
                      --database-resource "host=/var/run/postgresql user=$USER dbname=  
→$DBNAME sslmode=disable" \  
                      --ssl ../etc/Mosek/ssl \  
                      --mode gui
```

The Web GUI is not be available in HTTP mode, i.e. without encryption. The Web GUI provides role management, in particular creating users and generating their access tokens. When submitting a job the user should provide their access token through the parameter *MSK_SPAR_REMOTE_ACCESS_TOKEN* in the task.

To enable job submission by anonymous users specify `--enable-anonymous` in the setup step.

Chapter 7

REST API tutorials

This section contains tutorials for the OptServer REST API. Note that this should not be necessary in typical applications, where invoking the OptServer directly through the **MOSEK** API as discussed in [Sec. 5.3](#) is easier and more natural. Examples of that usage can be found in the manual for the respective Optimizer API (C, .NET, Python, Java) or Fusion API (C++, .NET, Python, Java).

- *Synchronous optimization tutorial*
 - problem submission,
 - solving and retrieving the result,
 - retrieving the solver log.
- *Asynchronous optimization tutorial*
 - problem submission,
 - solving,
 - checking if the solution is available,
 - retrieving the solver log in chunks,
 - retrieving the solution,
 - stopping the solver.

7.1 Synchronous Problem Submission

For the purpose of the tutorial we assume that the problem to be solved is read from a file, and the solutions will be saved to a file, i.e. we don't go into the logic which sets up the problem and interprets the solution. See [file formats](#) for specifications of file formats.

We demonstrate synchronous optimization, see [Sec. 5.1](#). If using authentication, the access token will always be passed in the **X-Mosek-Access-Token** header. Assuming that an HTTP/HTTPS connection to the OptServer was established, we first submit a problem using *submit*. We can provide a name for easier identification of the problem in the Web GUI. The file format is passed in the **Content-Type** header.

Listing 7.1: Submit a problem.

```
# POST problem data
con.request('POST', '/api/submit?jobname=' + jobname,
            probdata,
            headers = dict(headers, **{"Content-Type":
→ "application/x-mosek-{}".format(intype)}))
resp = con.getresponse()
check_status(resp)
# Recover a token identifying the job
token = resp.read().decode('ascii')
```

The response contains a token used to identify the job in future requests. If no errors have occurred, we use `solve` to request running the solver for the given job token. When requesting the solution we set the `Accept` header to indicate expected solution format.

Listing 7.2: Starting the solver synchronously.

```
# Use token to identify job
con.request('GET', '/api/solve?token=' + token,
            headers = dict(headers, **{"Accept": outtype}))
resp = con.getresponse()
```

The request will return when optimization terminates. If there were no errors, the status codes are available in the headers and the solution in the body of the response.

Listing 7.3: Retrieving results.

```
## Retrieve status codes
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)

## Retrieve the solution
if resp.status == http.client.OK:
    print('Solution (as plain text):\n')
    print(resp.read().decode('ascii', errors = 'ignore'))
```

It is also possible to retrieve the log from the solver (*log*):

Listing 7.4: Retrieving optimization log.

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

The full example is shown below.

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

```
import http.client
import sys
try:
    import ssl
except:
    pass

# A debug method which prints out the HTTP(S) response
# and exits in case of error
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]:
        raise Exception('An error in connection')

# A helper method which opens a connection
def openConnection(host, port, useHttps):
    if useHttps:
        return http.client.HTTPSConnection(host, port, context=ssl._create_unverified_
↪context())
```

(continues on next page)

```

else:
    return http.client.HTTPConnection(host, port)

# Main code
if __name__ == '__main__':
    try:
        # OptServer address
        host, port = sys.argv[1], int(sys.argv[2])
        # Protocol (HTTP/HTTPS)
        useHttps = (sys.argv[3] == "HTTPS")
        # Name of file with input data
        probfile = sys.argv[4]
        # Input and output file type
        intype, outtype = sys.argv[5], sys.argv[6]
        # Jobname (for demonstration)
        jobname = sys.argv[7]
        # Authentication token
        headers = {}
        if len(sys.argv) == 9:
            headers = {"X-Mosek-Access-Token": sys.argv[8]}
    except:
        print("Usage : python3 test_sync.py host          port protocol probfile_
↪intype outtype      jobname      [accestoken]")
        print("Example: python3 test_sync.py solve.mosek.com 38000 HTTPS      lo1.mps _
↪mps      text/plain SimpleTask ...")
        sys.exit(1)

    # Establish HTTP connection
    con = openConnection(host, port, useHttps)
    try:
        # Read input file
        with open(probfile, 'rb') as probdata:
            ## Submit job
            print('POST /api/submit')
            # POST problem data
            con.request('POST', '/api/submit?jobname=' + jobname,
                        probdata,
                        headers = dict(headers, **{"Content-Type":
↪"application/x-mosek-{}".format(intype)}))
            resp = con.getresponse()
            check_status(resp)
            # Recover a token identifying the job
            token = resp.read().decode('ascii')

            ## Solve and wait for solution
            print('GET /api/solve')
            # Use token to identify job
            con.request('GET', '/api/solve?token=' + token,
                        headers = dict(headers, **{"Accept": outtype}))
            resp = con.getresponse()
            check_status(resp)

            ## Retrieve status codes
            res = resp.getheader('X-Mosek-Res-Code', None)
            trm = resp.getheader('X-Mosek-Trm-Code', None)
            print('\tMOSEK response: %s' % res)

```

(continued from previous page)

```
print('\t\t\t term resp: %s' % term)

## Retrieve the solution
if resp.status == http.client.OK:
    print('Solution (as plain text):\n')
    print(resp.read().decode('ascii', errors = 'ignore'))

## Obtain the solver log output
print('GET /api/log')
con.request('GET', '/api/log?token=' + token, headers = headers)
resp = con.getresponse()
check_status(resp)
if resp.status == http.client.OK:
    print('Solver log:\n')
    print(resp.read().decode('utf-8', errors = 'ignore'))

finally:
    con.close()
```

It is also possible to simplify the logic by using the *submit+solve* API to submit the problem and receive the solution in one request. The remaining protocol is similar to the above so we restrict to listing the full example without further explanations.

Listing 7.6: How to submit a job and solve the problem synchronously in one request.

```
import http.client
import sys
try:
    import ssl
except:
    pass

# A debug method which prints out the HTTP(S) response
# and exits in case of error
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]:
        raise Exception('An error in connection')

# A helper method which opens a connection
def openConnection(host, port, useHttps):
    if useHttps:
        return http.client.HTTPSConnection(host, port, context=ssl._create_unverified_
context())
    else:
        return http.client.HTTPConnection(host, port)

# Main code
if __name__ == '__main__':
    try:
        # OptServer address
        host, port = sys.argv[1], int(sys.argv[2])
        # Protocol (HTTP/HTTPS)
        useHttps = (sys.argv[3] == "HTTPS")
```

(continues on next page)

```

# Name of file with input data
probfile = sys.argv[4]
# Input and output file type
intype, outtype = sys.argv[5], sys.argv[6]
# Jobname (for demonstration)
jobname = sys.argv[7]
# Authentication token
headers = {}
if len(sys.argv) == 9:
    headers = {"X-Mosek-Access-Token": sys.argv[8]}
except:
    print("Usage : python3 test_simple_api.py host          port protocol \
↪probfile intype outtype      jobname      [accestoken]")
    print("Example: python3 test_simple_api.py solve.mosek.com 38000 HTTPS \
↪lo1.mps mps      text/plain SimpleTask ...")
    sys.exit(1)

# Establish HTTP connection
con = openConnection(host, port, useHttps)
try:
    # Read input file
    with open(probfile, 'rb') as probdata:
        ## Submit job
        print('POST /api/submit+solve')
        # POST problem data
        con.request('POST', '/api/submit+solve?jobname=' + jobname,
                    probdata,
                    headers = dict(headers, **{"Content-Type":
↪"application/x-mosek-{}".format(intype),
                                                "Accept": outtype}))

        resp = con.getresponse()
        check_status(resp)
        # Recover a token identifying the job
        token = resp.getheader('X-Mosek-Token', None)

        ## Retrieve status codes
        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)

        ## Retrieve the solution
        if resp.status == http.client.OK:
            print('Solution (as plain text):\n')
            print(resp.read().decode('ascii', errors = 'ignore'))

        ## Obtain the solver log output
        print('GET /api/log')
        con.request('GET', '/api/log?token=' + token, headers = headers)
        resp = con.getresponse()
        check_status(resp)
        if resp.status == http.client.OK:
            print('Solver log:\n')
            print(resp.read().decode('utf-8', errors = 'ignore'))
finally:
    con.close()

```

7.2 Asynchronous Problem Submission

This tutorial demonstrates most features of the OptServer API, that is submitting a problem, polling for solution, retrieving the solution, breaking the solver and retrieving the log output.

For the purpose of the tutorial we assume that the problem to be solved is read from a file, and the solutions will be saved to a file, i.e. we don't go into the logic which sets up the problem and interprets the solution. See *file formats* for specifications of file formats.

Starting the solver

If using authentication, the access token will always be passed in the **X-Mosek-Access-Token** header. Assuming that an HTTP/HTTPS connection to the OptServer was established, we first submit a problem using *submit*. We can provide a name for easier identification of the problem in the Web GUI. The file format is passed in the **Content-Type** header.

Listing 7.7: Submit a problem.

```
# POST problem data
con.request('POST', '/api/submit?jobname=' + jobname,
            probdata,
            headers = dict(headers, **{"Content-Type":
↪ "application/x-mosek-{}".format(intype)}))
resp = con.getresponse()
check_status(resp)
# Recover a token identifying the job
token = resp.read().decode('ascii')
```

The response contains a token used to identify the job in future requests. Note that this operation is identical to the *synchronous case*. If no errors have occurred, we use *solve-background* to initiate solving the problem identified by the token:

Listing 7.8: Start solving the submission.

```
con.request("GET", "/api/solve-background?token=" + token, headers = {}
↪ headers)
resp = con.getresponse()
```

The calling program regains control immediately.

Waiting for and retrieving the solution

We can now periodically start polling for the solution via *solution*. We set the **Accept** header to indicate expected solution format. If the response is empty then the solution is not yet available:

Listing 7.9: Polling for the solution.

```
pollCount += 1
print("GET /api/solution")
con.request("GET", "/api/solution?token=" + token,
            headers = dict(headers, **{"Accept": outtype}))
resp = con.getresponse()
check_status(resp)

# Is the solution available?
if resp.status == http.client.NO_CONTENT:
    print("Solution not available in poll %d, continuing" % pollCount)
    time.sleep(1.0)
```

When the response becomes non-empty we can retrieve the solution:

Listing 7.10: Retrieving the solution when available.

```
elif resp.status == http.client.OK:
    solved = True
    res = resp.getheader('X-Mosek-Res-Code', None)
    trm = resp.getheader('X-Mosek-Trm-Code', None)
    print("\tMOSEK response: %s" % res)
    print("\t\t\ttrm resp: %s" % trm)

    print("Solution (as plain text):")
    print(resp.read().decode('ascii', errors = 'ignore'))
```

Stopping the solver

At some point we can decide that the optimization should be stopped. That can be done with *break*.

Listing 7.11: Stopping the solver.

```
if not solved and pollCount >= maxPolls:
    con = openConnection(host, port, useHttps)
    print("GET /api/break")
    con.request("GET", "/api/break?token=" + token, headers = headers)
    resp = con.getresponse()
    check_status(resp)
    con.close()
```

Note that the solver need not break immediately, in particular it can enter a few more loops of checking for solution. The **MOSEK** termination code in this case will be *MSK_RES_TRM_USER_CALLBACK*.

Retrieving the log

The log output from the solver can be retrieved gradually in each polling loop. The caller needs to keep track of how much of the log was already read and provide it as an offset in a call to *log*.

Listing 7.12: Retrieving log output.

```

con = openConnection(host, port, useHttps)
print("GET /api/log")
con.request("GET", "/api/log?token={0}&offset={1}".format(token, logOffset),
↳headers = headers)
resp = con.getResponse()
check_status(resp)
# Show the latest log entries
lastLog = resp.read().decode('ascii', errors = 'ignore')
print(lastLog)
# Update the log offset by the size received
logOffset += len(lastLog)
con.close()

```

Complete code

The full example is shown below.

Listing 7.13: How to submit a job and solve the problem asynchronously.

```

import http.client
import sys, time
try:
    import ssl
except:
    pass

# A debug method which prints out the HTTP(S) response
# and exits in case of error
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]:
        raise Exception('An error in connection')

# A helper method which opens a connection
def openConnection(host, port, useHttps):
    if useHttps:
        return http.client.HTTPSConnection(host, port, context=ssl._create_unverified_
↳context())
    else:
        return http.client.HTTPConnection(host, port)

if __name__ == '__main__':
    try:
        # OptServer address
        host, port = sys.argv[1], int(sys.argv[2])
        # Protocol (HTTP/HTTPS)
        useHttps = (sys.argv[3] == "HTTPS")
        # Name of file with input data
        probfile = sys.argv[4]
        # Input and output file type
        intype, outtype = sys.argv[5], sys.argv[6]

```

(continues on next page)

```

    # Number of solution polls
    maxPolls = int(sys.argv[7])
    # Jobname (for demonstration)
    jobname = sys.argv[8]
    # Authentication token
    headers = {}
    if len(sys.argv) == 10:
        headers = {"X-Mosek-Access-Token": sys.argv[9]}
    except:
        print("Usage : python3 test_async.py host          port  protocol  \
↪probfile intype outtype      maxPolls jobname      [accestoken]")
        print("Example: python3 test_async.py solve.mosek.com 38000 HTTPS      lo1.mps\
↪ mps      text/plain 5          SimpleTask  ...")
        sys.exit(1)

    token = ""

    # Create a connection for problem submission
    con = openConnection(host, port, useHttps)
    try:
        with open(probfile, 'rb') as probdata:
            ## Submit job
            print('POST /api/submit')
            # POST problem data
            con.request('POST', '/api/submit?jobname=' + jobname,
                        probdata,
                        headers = dict(headers, **{"Content-Type":
↪"application/x-mosek-{}".format(intype)}))
            resp = con.getresponse()
            check_status(resp)
            # Recover a token identifying the job
            token = resp.read().decode('ascii')

            ## Start solving and close connection
            print("GET /api/solve-background")
            con.request("GET", "/api/solve-background?token=" + token, headers = \
↪headers)
            resp = con.getresponse()
            check_status(resp)
        finally:
            con.close()
            print("Submit connection closed")

    # Begin waiting for the solution
    solved = False
    pollCount = 0
    logOffset = 0

    while not solved:
        con = openConnection(host, port, useHttps)
        pollCount += 1
        print("GET /api/solution")
        con.request("GET", "/api/solution?token=" + token,
                    headers = dict(headers, **{"Accept": outtype}))
        resp = con.getresponse()
        check_status(resp)

```

```

# Is the solution available?
if resp.status == http.client.NO_CONTENT:
    print("Solution not available in poll %d, continuing" % pollCount)
    time.sleep(1.0)
elif resp.status == http.client.OK:
    solved = True
    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)

    print("Solution (as plain text):")
    print(resp.read().decode('ascii', errors = 'ignore'))
con.close()

# After too many tries we indicate the solver to stop
if not solved and pollCount >= maxPolls:
    con = openConnection(host, port, useHttps)
    print("GET /api/break")
    con.request("GET", "/api/break?token=" + token, headers = headers)
    resp = con.getresponse()
    check_status(resp)
    con.close()

# Demonstrate how to retrieve the last part of solver log
con = openConnection(host, port, useHttps)
print("GET /api/log")
con.request("GET", "/api/log?token={0}&offset={1}".format(token, logOffset),
headers = headers)
resp = con.getresponse()
check_status(resp)
# Show the latest log entries
lastLog = resp.read().decode('ascii', errors = 'ignore')
print(lastLog)
# Update the log offset by the size received
logOffset += len(lastLog)
con.close()

```

A complete reference for the REST API can be found in [Sec. 9.1](#).

Chapter 8

Web GUI interface

The OptServer Web interface provides simple tools for monitoring and terminating jobs and managing users and their permissions and access tokens.

8.1 Setup

To activate the OptServer in GUI mode provide SSL keys and use the option `--mode gui` in the configuration process. See [Sec. 6.3](#) for details and an example. The Web GUI is only available with HTTPS enabled.

Before first login it is necessary to initialize the password for the `admin` user. Follow [Sec. 4.4.3](#).

Assuming the server is running and listening on host `HOST` and port `PORT`, the Web GUI login page is available at `https://HOST:PORT/`.

8.2 Users

A user with administrator privileges can manage users. Each user is created with a set of permissions to perform various actions:

- to log in,
- to use the API (require to be able to submit jobs),
- to submit jobs,
- to use authentication tokens,
- to create tokens,
- to perform administrator functions.

8.3 Tokens

Each user can be assigned authentication token(s). They can be used to identify the user through the API, for example when submitting jobs (see [Sec. 9.1.2](#)). Each token is created with an expiry date. If the validity period extends the value of `AccessTokenMaxExpiry` from the configuration file `server.conf`, then it will be truncated down to that value. If the permissions associated with the token exceed the permissions of the user, they will be reduced to match the permissions of the user. The users with permission to create tokens can also create tokens for themselves.

Users

Add user

Login:

Name:

Email:

Password:

Repeat Password:

Permissions:

- ☐ Login
- ☐ Submit
- ☐ Use API
- ☐ Use tokens
- ☐ Create tokens
- ☐ Administrator

Create

User ID	Name	Email	Password	Login	Submit	Use/Acc	Crt/Acc	API	Admin
admin	admin		****	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
alice	Alice	alice@mosek.com	****	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
bob	Bob	bob@mosek.com	****	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
eve	Eve	eve@mosek.com	****	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Fig. 8.1: A screenshot of the user page.

All Access Tokens

Create new Token

User:

Description:

Expiry date:

☒ Can submit

☒ Administrate

Create access token

Generated Token:

Selection

Delete selected

Description	Owner	Expiry	Submit	Admin
Bob token 1	bob	41 days, 14 hours	yes	no
Eve for testing	eve	10 days, 14 hours	yes	no
Bob everything	bob	36 days, 14 hours	yes	yes

Fig. 8.2: A screenshot of the tokens page.

8.4 Jobs

The jobs page gives access to a record of all jobs executed on the server, together with some basic data like job ID, description, owner, log output etc. A simple search engine to filter the jobs is available.

All Jobs

Selection

Jobs

Job ID	Description	Owner	Status	Submitted	T	Sz	
124395259606e9b31c06ce7bce240368		eve	Completed	2019-09-24 12:54:56	0:0:0.02	283 B	i ▶ ■ ☹
496ccf655e7fce6bb7cdba640a76a055	TestJob	eve	Completed	2019-09-24 13:05:50	0:0:0.02	283 B	i ▶ ■ ☹
8ab99c27b02a4a84aac42ac352624581	LeastSquares1	eve	Completed	2019-09-24 13:06:44	0:0:0.03	15.80 KB	i ▶ ■ ☹
005fa7a8d00c11c6dc3d670923b4ecab	LeastSquares2	eve	Completed	2019-09-24 13:15:47	0:0:0.03	15.80 KB	i ▶ ■ ☹
4443e41f9808e6c036130698b0f0546b	BobRegression1	bob	Completed	2019-09-24 13:16:17	0:0:0.04	15.80 KB	i ▶ ■ ☹

Job ID	Description	Owner	Status	Submitted	T	Sz
--------	-------------	-------	--------	-----------	---	----

Fig. 8.3: A screenshot of the jobs page.

8.5 User's profile

Each user can see and edit their own data in the rightmost menu item. That menu gives also access to the user's tokens page and the user's jobs page, which are analogous to the global ones described earlier.

Chapter 9

OptServer Reference

- *REST API Protocol specification*
- **Optimizer parameters:**
 - *Double, Integer, String*
 - *Full list*
 - *Browse by topic*
- *Optimizer response codes*

9.1 OptServer REST API

9.1.1 Commands

This section describes the REST API of the OptServer. Additional authentication options, common to all commands, are described in [Sec. 9.1.2](#).

POST `/api/submit`

Submit a problem to the server.

The problem file should be submitted in the content of the request. The **Content-Type** header should specify the file format of the submission (if not present, the solver may guess incorrect format and fail to start the solver). The recognized content types are listed in [Table 9.1](#).

Table 9.1: Content types in submit.

Content-Type	File format
application/x-mosek-task	MOSEK Task
application/x-mosek-json	MOSEK JTask (JSON)
application/x-mosek-lp	LP format
application/x-mosek-mps	MPS format
application/x-mosek-opf	OPF format
application/x-mosek-cbf	CBF format
application/x-mosek-ptf	PTF format
application/x-mosek-XXX+gzip	XXX format compressed with gzip
application/x-mosek-XXX+zstd	XXX format compressed with zstd

See [Sec. 10](#) for descriptions of supported formats.

The name of the job can be specified in a query string `jobname=...`

On response OK a token identifying the problem is returned in the response body, in the session cookie and in the header **X-Mosek-Token**. That token is required to identify the job in future request.

GET /api/solve

Start solving and wait for the solver to finish.

The job to start is specified in the query string `token=...`

The file format of the solution can be specified in the **Accept** header (if not present, a plain text ASCII solution will be returned), as in Table 9.2.

The solution is returned as the content of the response and the headers are set as in Table 9.3.

POST /api/submit+solve

Submit a problem to the server and get the solution.

Performs the actions of **submit** and **solve** in one request. The problem file should be submitted in the content of the request. The **Content-Type** header should specify the file format of the submission (if not present, the solver may guess incorrect format and fail to start the solver). The recognized content types are listed in Table 9.1.

See Sec. 10 for descriptions of supported formats.

The file format of the solution can be specified in the **Accept** header (if not present, a plain text ASCII solution will be returned), as in Table 9.2. The solution is returned as the content of the response and the headers are set as in Table 9.3.

The name of the job can be specified in a query string `jobname=...`

On response OK a token identifying the problem is returned in the header **X-Mosek-Token**. That token is required to identify the job in future request.

GET /api/solve-background

Start solving in the background and return immediately.

The job to start is specified in the query string `token=...`

It returns OK if the solver started successfully.

GET /api/solution

Return the solution

The problem whose solution is requested is specified in the query string `token=...`

The file format of the solution can be specified in the **Accept** header (if not present, a plain text ASCII solution will be returned). The recognized types are:

Table 9.2: Accepted solution formats in solution and solve.

Accept	Solution format
application/x-mosek-task	MOSEK Task
application/x-mosek-json	MOSEK JSol file (JSON)
application/json	MOSEK JSol file (JSON)
text/plain	Plain text

See Sec. 10 for descriptions of supported solution formats.

If the solution is not yet available, the call returns an empty response with no content.

If the solution is available it is returned as the content of the response and the following headers are set:

Table 9.3: Headers set in the response to solution and solve.

Header	Value
Content-Type	Solution type as requested in Accept
X-Mosek-Res-Code	Response code from the optimizer
X-Mosek-Trm-Code	Termination code from the optimizer
X-Mosek-Token	Job token
Content-Length	Length of the solution

If an unexpected error occurred then **X-Mosek-Res-Code** will be set to **MSK_RES_UNKNOWN** and the other fields are not defined.

GET /api/log

Return the log.

The problem for which the log output is requested is specified in the query string `token=...`

If the query string contains the parameter `offset=XXXX`, the log file will be returned from offset `XXXX` until the end of what is currently available. Otherwise the whole log is returned.

GET /api/break

Attempt to terminate the solver.

The problem to be terminated is specified in the query string `token=...`

9.1.2 Authentication

If the OptServer allows anonymous job submission then no authentication is required. Otherwise all of the commands require authentication in one of the following ways:

- The user's access token is passed as a query string `access-token=...` in the request.
- The user's access token is passed in the header `X-Mosek-Access-Token` of the request.
- Through a session cookie, if the user has logged in and authenticated within an open session.

Access tokens for users can be generated through the Web GUI, see [Sec. 8](#).

9.2 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_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_WRITE_HEADER*
- *MSK_IPAR_OPF_WRITE_HINTS*
- *MSK_IPAR_OPF_WRITE_LINE_LENGTH*
- *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_PTF_WRITE_TRANSFORM*
- *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_COMPRESSION*
- *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_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_GP_NUM_SEEDS*
- *MSK_IPAR_INTPNT_ORDER_METHOD*
- *MSK_IPAR_INTPNT_PURIFY*
- *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_INCLUDE_SUMMARY*
- *MSK_IPAR_LOG_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_LOCAL_INFO*
- *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_MAX_TIME*
- *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_CONIC_OUTER_APPROXIMATION*
- *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_FEASPUMP_LEVEL*
- *MSK_IPAR_MIO_HEURISTIC_LEVEL*
- *MSK_IPAR_MIO_MAX_NUM_BRANCHES*
- *MSK_IPAR_MIO_MAX_NUM_RELAXS*
- *MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS*
- *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_PROPAGATE_OBJECTIVE_CONSTRAINT*
- *MSK_IPAR_MIO_RINS_MAX_NODES*
- *MSK_IPAR_MIO_ROOT_OPTIMIZER*
- *MSK_IPAR_MIO_ROOT_REPEAT_PREOLVE_LEVEL*
- *MSK_IPAR_MIO_SEED*
- *MSK_IPAR_MIO_VB_DETECTION_LEVEL*

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_INCLUDE_SUMMARY*
- *MSK_IPAR_LOG_INFEAS_ANA*
- *MSK_IPAR_LOG_INTPNT*
- *MSK_IPAR_LOG_LOCAL_INFO*
- *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_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_PASS*
- *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_SEED*
- *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_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_MAX_TIME*
- *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_ROOT_CUT_ROUNDS*
- *MSK_IPAR_MIO_MAX_NUM_SOLUTIONS*
- *MSK_IPAR_SIM_MAX_ITERATIONS*

Other

- *MSK_IPAR_COMPRESS_STATFILE*

9.3 Parameters (alphabetical list sorted by type)

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

9.3.1 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]

Example mosek -d MSK_DPAR_ANA_SOL_INFEAS_TOL 1e-6 file

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]

Example mosek -d MSK_DPAR_BASIS_REL_TOL_S 1.0e-12 file

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]

Example mosek -d MSK_DPAR_BASIS_TOL_S 1.0e-6 file

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]

Example mosek -d MSK_DPAR_BASIS_TOL_X 1.0e-6 file

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:

$$d_i \leq -|Q_{ii}| \text{check_convexity_rel_tol}$$

Default 1e-10

Accepted [0; +inf]

Example `mosek -d MSK_DPAR_CHECK_CONVEXITY_REL_TOL 1e-10 file`

Groups *Interior-point method*

MSK_DPAR_DATA_SYM_MAT_TOL

Absolute zero tolerance for elements in symmetric matrices. 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]

Example `mosek -d MSK_DPAR_DATA_SYM_MAT_TOL 1.0e-12 file`

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]

Example `mosek -d MSK_DPAR_DATA_SYM_MAT_TOL_HUGE 1.0e20 file`

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]

Example `mosek -d MSK_DPAR_DATA_SYM_MAT_TOL_LARGE 1.0e10 file`

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]

Example `mosek -d MSK_DPAR_DATA_TOL_AIJ_HUGE 1.0e20 file`

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]

Example `mosek -d MSK_DPAR_DATA_TOL_AIJ_LARGE 1.0e10 file`

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]

Example `mosek -d MSK_DPAR_DATA_TOL_BOUND_INF 1.0e16 file`

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]

Example `mosek -d MSK_DPAR_DATA_TOL_BOUND_WRN 1.0e8 file`

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]

Example `mosek -d MSK_DPAR_DATA_TOL_C_HUGE 1.0e16 file`

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]

Example `mosek -d MSK_DPAR_DATA_TOL_CJ_LARGE 1.0e8 file`

Groups *Data check*

MSK_DPAR_DATA_TOL_QIJ

Absolute zero tolerance for elements in Q matrices.

Default 1.0e-16

Accepted [0.0; +inf]

Example `mosek -d MSK_DPAR_DATA_TOL_QIJ 1.0e-16 file`

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]

Example `mosek -d MSK_DPAR_DATA_TOL_X 1.0e-8 file`

Groups *Data check*

MSK_DPAR_INTPNT_CO_TOL_DFEAS

Dual feasibility tolerance used by the interior-point optimizer for conic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_CO_TOL_DFEAS 1.0e-8 file`

See also *MSK_DPAR_INTPNT_CO_TOL_NEAR_REL*

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

MSK_DPAR_INTPNT_CO_TOL_INFEAS

Infeasibility tolerance used by the interior-point optimizer for conic problems. Controls when the 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-12

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_CO_TOL_INFEAS 1.0e-12 file`

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

MSK_DPAR_INTPNT_CO_TOL_MU_RED

Relative complementarity gap tolerance used by the interior-point optimizer for conic problems.

Default 1.0e-8
Accepted [0.0; 1.0]
Example mosek -d MSK_DPAR_INTPNT_CO_TOL_MU_RED 1.0e-8 file
Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_NEAR_REL

Optimality tolerance used by the interior-point optimizer for conic problems. If **MOSEK** cannot compute a solution that has the prescribed accuracy then it will check if the solution found satisfies the termination criteria with all tolerances multiplied by the value of this parameter. If yes, then the solution is also declared optimal.

Default 1000
Accepted [1.0; +inf]
Example mosek -d MSK_DPAR_INTPNT_CO_TOL_NEAR_REL 1000 file
Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_PFEAS

Primal feasibility tolerance used by the interior-point optimizer for conic problems.

Default 1.0e-8
Accepted [0.0; 1.0]
Example mosek -d MSK_DPAR_INTPNT_CO_TOL_PFEAS 1.0e-8 file
See also [MSK_DPAR_INTPNT_CO_TOL_NEAR_REL](#)
Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_CO_TOL_REL_GAP

Relative gap termination tolerance used by the interior-point optimizer for conic problems.

Default 1.0e-8
Accepted [0.0; 1.0]
Example mosek -d MSK_DPAR_INTPNT_CO_TOL_REL_GAP 1.0e-8 file
See also [MSK_DPAR_INTPNT_CO_TOL_NEAR_REL](#)
Groups *Interior-point method, Termination criteria, Conic interior-point method*

MSK_DPAR_INTPNT_QO_TOL_DFEAS

Dual feasibility tolerance used by the interior-point optimizer for quadratic problems.

Default 1.0e-8
Accepted [0.0; 1.0]
Example mosek -d MSK_DPAR_INTPNT_QO_TOL_DFEAS 1.0e-8 file
See also [MSK_DPAR_INTPNT_QO_TOL_NEAR_REL](#)
Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_INFEAS

Infeasibility tolerance used by the interior-point optimizer for quadratic problems. Controls when the 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-12
Accepted [0.0; 1.0]
Example mosek -d MSK_DPAR_INTPNT_QO_TOL_INFEAS 1.0e-12 file
Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_MU_RED

Relative complementarity gap tolerance used by the interior-point optimizer for quadratic problems.

Default 1.0e-8
Accepted [0.0; 1.0]
Example mosek -d MSK_DPAR_INTPNT_QO_TOL_MU_RED 1.0e-8 file

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_NEAR_REL

Optimality tolerance used by the interior-point optimizer for quadratic problems. If **MOSEK** cannot compute a solution that has the prescribed accuracy then it will check if the solution found satisfies the termination criteria with all tolerances multiplied by the value of this parameter. If yes, then the solution is also declared optimal.

Default 1000

Accepted [1.0; +inf]

Example `mosek -d MSK_DPAR_INTPNT_QO_TOL_NEAR_REL 1000 file`

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_PFEAS

Primal feasibility tolerance used by the interior-point optimizer for quadratic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_QO_TOL_PFEAS 1.0e-8 file`

See also [*MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*](#)

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_QO_TOL_REL_GAP

Relative gap termination tolerance used by the interior-point optimizer for quadratic problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_QO_TOL_REL_GAP 1.0e-8 file`

See also [*MSK_DPAR_INTPNT_QO_TOL_NEAR_REL*](#)

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_TOL_DFEAS

Dual feasibility tolerance used by the interior-point optimizer for linear problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_TOL_DFEAS 1.0e-8 file`

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]

Example `mosek -d MSK_DPAR_INTPNT_TOL_DSAFE 1.0 file`

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_INFEAS

Infeasibility tolerance used by the interior-point optimizer for linear problems. Controls when the 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]

Example `mosek -d MSK_DPAR_INTPNT_TOL_INFEAS 1.0e-10 file`

Groups *Interior-point method, Termination criteria*

MSK_DPAR_INTPNT_TOL_MU_RED

Relative complementarity gap tolerance used by the interior-point optimizer for linear problems.

Default 1.0e-16

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_TOL_MU_RED 1.0e-16 file`

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 path is followed very closely. On numerically unstable problems it may be worthwhile to increase this parameter.

Default 1.0e-8

Accepted [0.0; 0.9999]

Example `mosek -d MSK_DPAR_INTPNT_TOL_PATH 1.0e-8 file`

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_PFEAS

Primal feasibility tolerance used by the interior-point optimizer for linear problems.

Default 1.0e-8

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_TOL_PFEAS 1.0e-8 file`

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]

Example `mosek -d MSK_DPAR_INTPNT_TOL_PSAFE 1.0 file`

Groups *Interior-point method*

MSK_DPAR_INTPNT_TOL_REL_GAP

Relative gap termination tolerance used by the interior-point optimizer for linear problems.

Default 1.0e-8

Accepted [1.0e-14; +inf]

Example `mosek -d MSK_DPAR_INTPNT_TOL_REL_GAP 1.0e-8 file`

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]

Example `mosek -d MSK_DPAR_INTPNT_TOL_REL_STEP 0.9999 file`

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 to stop.

Default 1.0e-6

Accepted [0.0; 1.0]

Example `mosek -d MSK_DPAR_INTPNT_TOL_STEP_SIZE 1.0e-6 file`

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]

Example `mosek -d MSK_DPAR_LOWER_OBJ_CUT -1.0e30 file`

See also *MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH*

Groups *Termination criteria*

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]

Example `mosek -d MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH -0.5e30 file`

Groups *Termination criteria*

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]

Example `mosek -d MSK_DPAR_MIO_MAX_TIME -1.0 file`

Groups *Mixed-integer optimization, Termination criteria*

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]

Example `mosek -d MSK_DPAR_MIO_REL_GAP_CONST 1.0e-10 file`

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]

Example `mosek -d MSK_DPAR_MIO_TOL_ABS_GAP 0.0 file`

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]

Example `mosek -d MSK_DPAR_MIO_TOL_ABS_RELAX_INT 1.0e-5 file`

Groups *Mixed-integer optimization*

MSK_DPAR_MIO_TOL_FEAS

Feasibility tolerance for mixed integer solver.

Default 1.0e-6

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

Example mosek -d MSK_DPAR_MIO_TOL_FEAS 1.0e-6 file

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]

Example mosek -d MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT 0.0 file

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]

Example mosek -d MSK_DPAR_MIO_TOL_REL_GAP 1.0e-4 file

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]

Example mosek -d MSK_DPAR_OPTIMIZER_MAX_TIME -1.0 file

Groups *Termination criteria*

MSK_DPAR_PREOLVE_TOL_ABS_LINDEP

Absolute tolerance employed by the linear dependency checker.

Default 1.0e-6

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_ABS_LINDEP 1.0e-6 file

Groups *Presolve*

MSK_DPAR_PREOLVE_TOL_AIJ

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

Default 1.0e-12

Accepted [1.0e-15; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_AIJ 1.0e-12 file

Groups *Presolve*

MSK_DPAR_PREOLVE_TOL_REL_LINDEP

Relative tolerance employed by the linear dependency checker.

Default 1.0e-10

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_REL_LINDEP 1.0e-10 file

Groups *Presolve*

MSK_DPAR_PREOLVE_TOL_S

Absolute zero tolerance employed for s_i in the presolve.

Default 1.0e-8

Accepted [0.0; +inf]

Example mosek -d MSK_DPAR_PREOLVE_TOL_S 1.0e-8 file

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]

Example mosek -d MSK_DPAR_PRESOLVE_TOL_X 1.0e-8 file

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]

Example mosek -d MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL 1e-15 file

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]

Example mosek -d MSK_DPAR_SEMIDEFINITE_TOL_APPROX 1.0e-10 file

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]

Example mosek -d MSK_DPAR_SIM_LU_TOL_REL_PIV 0.01 file

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]

Example mosek -d MSK_DPAR_SIMPLEX_ABS_TOL_PIV 1.0e-7 file

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]

Example mosek -d MSK_DPAR_UPPER_OBJ_CUT 1.0e30 file

See also *MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH*

Groups *Termination criteria*

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]
Example mosek -d MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH 0.5e30 file
Groups *Termination criteria*

9.3.2 Integer parameters

MSK_IPAR_ANA_SOL_BASIS

Controls whether the basis matrix is analyzed in solution analyzer.

Default *ON*
Accepted *ON, OFF*
Example mosek -d MSK_IPAR_ANA_SOL_BASIS MSK_ON file
Groups *Analysis*

MSK_IPAR_ANA_SOL_PRINT_VIOLATED

A parameter of the problem analyzer. 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*
Example mosek -d MSK_IPAR_ANA_SOL_PRINT_VIOLATED MSK_OFF file
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*
Example mosek -d MSK_IPAR_AUTO_SORT_A_BEFORE_OPT MSK_OFF file
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*
Example mosek -d MSK_IPAR_AUTO_UPDATE_SOL_INFO MSK_OFF file
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*
Example mosek -d MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE MSK_OFF file
Groups *Simplex optimizer*

MSK_IPAR_BI_CLEAN_OPTIMIZER

Controls which simplex optimizer is used in the clean-up phase. Anything else than *MSK_OPTIMIZER_PRIMAL_SIMPLEX* or *MSK_OPTIMIZER_DUAL_SIMPLEX* is equivalent to *MSK_OPTIMIZER_FREE_SIMPLEX*.

Default *FREE*

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

Example `mosek -d MSK_IPAR_BI_CLEAN_OPTIMIZER MSK_OPTIMIZER_FREE file`

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*

Example `mosek -d MSK_IPAR_BI_IGNORE_MAX_ITER MSK_OFF file`

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*

Example `mosek -d MSK_IPAR_BI_IGNORE_NUM_ERROR MSK_OFF file`

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]

Example `mosek -d MSK_IPAR_BI_MAX_ITERATIONS 1000000 file`

Groups *Basis identification, Termination criteria*

MSK_IPAR_CACHE_LICENSE

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

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

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_CACHE_LICENSE MSK_ON file`

Groups *License manager*

MSK_IPAR_CHECK_CONVEXITY

Specify the level of convexity check on quadratic problems.

Default *FULL*

Accepted *NONE, SIMPLE, FULL*

Example `mosek -d MSK_IPAR_CHECK_CONVEXITY MSK_CHECK_CONVEXITY_FULL file`

Groups *Data check*

MSK_IPAR_COMPRESS_STATFILE

Control compression of stat files.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_COMPRESS_STATFILE MSK_ON file

MSK_IPAR_INFEAS_GENERIC_NAMES

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

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_INFEAS_GENERIC_NAMES MSK_OFF file

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*

Example mosek -d MSK_IPAR_INFEAS_PREFER_PRIMAL MSK_ON file

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*

Example mosek -d MSK_IPAR_INFEAS_REPORT_AUTO MSK_OFF file

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]*

Example mosek -d MSK_IPAR_INFEAS_REPORT_LEVEL 1 file

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, RESERVED*

Example mosek -d MSK_IPAR_INTPNT_BASIS MSK_BI_ALWAYS file

See also *MSK_IPAR_BI_IGNORE_MAX_ITER, MSK_IPAR_BI_IGNORE_NUM_ERROR, MSK_IPAR_BI_MAX_ITERATIONS, MSK_IPAR_BI_CLEAN_OPTIMIZER*

Groups *Interior-point method, Basis identification*

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.

Example mosek -d MSK_IPAR_INTPNT_DIFF_STEP MSK_ON file

Groups *Interior-point method*

MSK_IPAR_INTPNT_HOTSTART

Currently not in use.

Default *NONE*

Accepted *NONE, PRIMAL, DUAL, PRIMAL_DUAL*

Example `mosek -d MSK_IPAR_INTPNT_HOTSTART MSK_INTPNT_HOTSTART_NONE file`

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]

Example `mosek -d MSK_IPAR_INTPNT_MAX_ITERATIONS 400 file`

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]

Example `mosek -d MSK_IPAR_INTPNT_MAX_NUM_COR -1 file`

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]

Example `mosek -d MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS -1 file`

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*

Example `mosek -d MSK_IPAR_INTPNT_MULTI_THREAD MSK_ON file`

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]

Example `mosek -d MSK_IPAR_INTPNT_OFF_COL_TRH 40 file`

Groups *Interior-point method*

MSK_IPAR_INTPNT_ORDER_GP_NUM_SEEDS

The GP ordering is dependent on a random seed. Therefore, trying several random seeds may lead to a better ordering. This parameter controls the number of random seeds tried.

A value of 0 means that MOSEK makes the choice.

Default 0

Accepted [0; +inf]

Example mosek -d MSK_IPAR_INTPNT_ORDER_GP_NUM_SEEDS 0 file

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*

Example mosek -d MSK_IPAR_INTPNT_ORDER_METHOD MSK_ORDER_METHOD_FREE
file

Groups *Interior-point method*

MSK_IPAR_INTPNT_PURIFY

Currently not in use.

Default *NONE*

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

Example mosek -d MSK_IPAR_INTPNT_PURIFY MSK_PURIFY_NONE file

Groups *Interior-point method*

MSK_IPAR_INTPNT_REGULARIZATION_USE

Controls whether regularization is allowed.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_INTPNT_REGULARIZATION_USE MSK_ON file

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*

Example mosek -d MSK_IPAR_INTPNT_SCALING MSK_SCALING_FREE file

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*

Example mosek -d MSK_IPAR_INTPNT_SOLVE_FORM MSK_SOLVE_FREE file

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*

Example mosek -d MSK_IPAR_INTPNT_STARTING_POINT MSK_STARTING_POINT_FREE
file

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*

Example mosek -d MSK_IPAR_LICENSE_DEBUG MSK_OFF file

Groups *License manager*

MSK_IPAR_LICENSE_PAUSE_TIME

If *MSK_IPAR_LICENSE_WAIT* is *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]

Example `mosek -d MSK_IPAR_LICENSE_PAUSE_TIME 100 file`

Groups *License manager*

MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS

Controls whether license features expire warnings are suppressed.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS MSK_OFF file`

Groups *License manager, Output information*

MSK_IPAR_LICENSE_TRH_EXPIRY_WRN

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

Default 7

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LICENSE_TRH_EXPIRY_WRN 7 file`

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*

Example `mosek -d MSK_IPAR_LICENSE_WAIT MSK_OFF file`

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]

Example `mosek -d MSK_IPAR_LOG 10 file`

See also *MSK_IPAR_LOG_CUT_SECOND_OPT*

Groups *Output information, Logging*

MSK_IPAR_LOG_ANA_PRO

Controls amount of output from the problem analyzer.

Default 1

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_ANA_PRO 1 file`

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]

Example mosek -d MSK_IPAR_LOG_BI 1 file

Groups *Basis identification, Output information, Logging*

MSK_IPAR_LOG_BI_FREQ

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

Default 2500

Accepted [0; +inf]

Example mosek -d MSK_IPAR_LOG_BI_FREQ 2500 file

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]

Example mosek -d MSK_IPAR_LOG_CHECK_CONVEXITY 0 file

Groups *Data check*

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]

Example mosek -d MSK_IPAR_LOG_CUT_SECOND_OPT 1 file

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

Groups *Output information, Logging*

MSK_IPAR_LOG_EXPAND

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

Default 0

Accepted [0; +inf]

Example mosek -d MSK_IPAR_LOG_EXPAND 0 file

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]

Example mosek -d MSK_IPAR_LOG_FEAS_REPAIR 1 file

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]

Example mosek -d MSK_IPAR_LOG_FILE 1 file

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

MSK_IPAR_LOG_INCLUDE_SUMMARY

Not relevant for this API.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_LOG_INCLUDE_SUMMARY MSK_OFF file

Groups *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]

Example mosek -d MSK_IPAR_LOG_INFEAS_ANA 1 file

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]

Example mosek -d MSK_IPAR_LOG_INTPNT 1 file

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

MSK_IPAR_LOG_LOCAL_INFO

Controls whether local identifying information like environment variables, filenames, IP addresses etc. are printed to the log.

Note that this will only affect some functions. Some functions that specifically emit system information will not be affected.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_LOG_LOCAL_INFO MSK_ON file

Groups *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]

Example mosek -d MSK_IPAR_LOG_MIO 4 file

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]

Example `mosek -d MSK_IPAR_LOG_MIO_FREQ 10 file`

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]

Example `mosek -d MSK_IPAR_LOG_ORDER 1 file`

Groups *Output information, Logging*

MSK_IPAR_LOG_PREOLVE

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

Default 1

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_LOG_PREOLVE 1 file`

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]

Example `mosek -d MSK_IPAR_LOG_RESPONSE 0 file`

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]

Example `mosek -d MSK_IPAR_LOG_SENSITIVITY 1 file`

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]

Example `mosek -d MSK_IPAR_LOG_SENSITIVITY_OPT 0 file`

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]

Example `mosek -d MSK_IPAR_LOG_SIM 4 file`

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]

Example mosek -d MSK_IPAR_LOG_SIM_FREQ 1000 file

Groups *Simplex optimizer, Output information, Logging*

MSK_IPAR_LOG_SIM_MINOR

Currently not in use.

Default 1

Accepted [0; +inf]

Example mosek -d MSK_IPAR_LOG_SIM_MINOR 1 file

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]

Example mosek -d MSK_IPAR_LOG_STORAGE 0 file

Groups *Output information, Overall system, Logging*

MSK_IPAR_MAX_NUM_WARNINGS

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

Default 10

Accepted [-inf; +inf]

Example mosek -d MSK_IPAR_MAX_NUM_WARNINGS 10 file

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*

Example mosek -d MSK_IPAR_MIO_BRANCH_DIR MSK_BRANCH_DIR_FREE file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION

If this option is turned on outer approximation is used when solving relaxations of conic problems; otherwise interior point is used.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION MSK_OFF file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CLIQUE

Controls whether clique cuts should be generated.

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_MIO_CUT_CLIQUE MSK_ON file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_CMIR

Controls whether mixed integer rounding cuts should be generated.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_CMIR MSK_ON file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_GMI

Controls whether GMI cuts should be generated.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_GMI MSK_ON file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_IMPLIED_BOUND

Controls whether implied bound cuts should be generated.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_IMPLIED_BOUND MSK_OFF file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_CUT_KNAPSACK_COVER

Controls whether knapsack cover cuts should be generated.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_CUT_KNAPSACK_COVER MSK_OFF file`

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]

Example `mosek -d MSK_IPAR_MIO_CUT_SELECTION_LEVEL -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_FEASPUMP_LEVEL

Controls the way the Feasibility Pump heuristic is employed by the mixed-integer optimizer.

- -1. The optimizer chooses how the Feasibility Pump is used
- 0. The Feasibility Pump is disabled
- 1. The Feasibility Pump is enabled with an effort to improve solution quality
- 2. The Feasibility Pump is enabled with an effort to reach feasibility early

Default -1

Accepted [-1; 2]

Example `mosek -d MSK_IPAR_MIO_FEASPUMP_LEVEL -1 file`

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]

Example mosek -d MSK_IPAR_MIO_HEURISTIC_LEVEL -1 file

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]

Example mosek -d MSK_IPAR_MIO_MAX_NUM_BRANCHES -1 file

Groups *Mixed-integer optimization, Termination criteria*

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]

Example mosek -d MSK_IPAR_MIO_MAX_NUM_RELAXS -1 file

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS

Maximum number of cut separation rounds at the root node.

Default 100

Accepted [0; +inf]

Example mosek -d MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS 100 file

Groups *Mixed-integer optimization, Termination criteria*

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]

Example mosek -d MSK_IPAR_MIO_MAX_NUM_SOLUTIONS -1 file

Groups *Mixed-integer optimization, Termination criteria*

MSK_IPAR_MIO_MODE

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

Default *SATISFIED*

Accepted *IGNORED, SATISFIED*

Example mosek -d MSK_IPAR_MIO_MODE MSK_MIO_MODE_SATISFIED file

Groups *Overall solver*

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*

Example `mosek -d MSK_IPAR_MIO_NODE_OPTIMIZER MSK_OPTIMIZER_FREE file`

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, PSEUDO*

Example `mosek -d MSK_IPAR_MIO_NODE_SELECTION MSK_MIO_NODE_SELECTION_FREE file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE

Enables or disables perspective reformulation in presolve.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE MSK_ON file`

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]*

Example `mosek -d MSK_IPAR_MIO_PROBING_LEVEL -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_PROPAGATE_OBJECTIVE_CONSTRAINT

Use objective domain propagation.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_MIO_PROPAGATE_OBJECTIVE_CONSTRAINT MSK_OFF file`

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]*

Example `mosek -d MSK_IPAR_MIO_RINS_MAX_NODES -1 file`

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*

Example `mosek -d MSK_IPAR_MIO_ROOT_OPTIMIZER MSK_OPTIMIZER_FREE file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_ROOT_REPEAT_PREOLVE_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]

Example `mosek -d MSK_IPAR_MIO_ROOT_REPEAT_PREOLVE_LEVEL -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MIO_SEED

Sets the random seed used for randomization in the mixed integer optimizer. Selecting a different seed can change the path the optimizer takes to the optimal solution.

Default 42

Accepted [0; +inf]

Example `mosek -d MSK_IPAR_MIO_SEED 42 file`

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]

Example `mosek -d MSK_IPAR_MIO_VB_DETECTION_LEVEL -1 file`

Groups *Mixed-integer optimization*

MSK_IPAR_MT_SPINCOUNT

Set the number of iterations to spin before sleeping.

Default 0

Accepted [0; 1000000000]

Example `mosek -d MSK_IPAR_MT_SPINCOUNT 0 file`

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.

If using the conic optimizer, the value of this parameter set at first optimization remains constant through the lifetime of the process. **MOSEK** will allocate a thread pool of given size, and changing the parameter value later will have no effect. It will, however, remain possible to demand single-threaded execution by setting *MSK_IPAR_INTPNT_MULTI_THREAD*.

For the mixed-integer optimizer and interior-point linear optimizer there is no such restriction.

On the `linuxaarch64` platform the settings only applies to the mixed-integer optimizer; the continuous optimizers are single-threaded.

Default 0
Accepted [0; +inf]
Example mosek -d MSK_IPAR_NUM_THREADS 0 file
Groups *Overall system*

MSK_IPAR_OPF_WRITE_HEADER

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

Default *ON*
Accepted *ON, OFF*
Example mosek -d MSK_IPAR_OPF_WRITE_HEADER MSK_ON file
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*
Example mosek -d MSK_IPAR_OPF_WRITE_HINTS MSK_ON file
Groups *Data input/output*

MSK_IPAR_OPF_WRITE_LINE_LENGTH

Aim to keep lines in OPF files not much longer than this.

Default 80
Accepted [0; +inf]
Example mosek -d MSK_IPAR_OPF_WRITE_LINE_LENGTH 80 file
Groups *Data input/output*

MSK_IPAR_OPF_WRITE_PARAMETERS

Write a parameter section in an OPF file.

Default *OFF*
Accepted *ON, OFF*
Example mosek -d MSK_IPAR_OPF_WRITE_PARAMETERS MSK_OFF file
Groups *Data input/output*

MSK_IPAR_OPF_WRITE_PROBLEM

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

Default *ON*
Accepted *ON, OFF*
Example mosek -d MSK_IPAR_OPF_WRITE_PROBLEM MSK_ON file
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*
Example mosek -d MSK_IPAR_OPF_WRITE_SOL_BAS MSK_ON file
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*

Example mosek -d MSK_IPAR_OPF_WRITE_SOL_ITG MSK_ON file

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*

Example mosek -d MSK_IPAR_OPF_WRITE_SOL_ITR MSK_ON file

Groups *Data input/output*

MSK_IPAR_OPF_WRITE_SOLUTIONS

Enable inclusion of solutions in the OPF files.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_OPF_WRITE_SOLUTIONS MSK_OFF file

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*

Example mosek -d MSK_IPAR_OPTIMIZER MSK_OPTIMIZER_FREE file

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*

Example mosek -d MSK_IPAR_PARAM_READ_CASE_NAME MSK_ON file

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*

Example mosek -d MSK_IPAR_PARAM_READ_IGN_ERROR MSK_OFF file

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]*

Example mosek -d MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_FILL -1 file

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]*

Example `mosek -d MSK_IPAR_PRESOLVE_ELIMINATOR_MAX_NUM_TRIES -1 file`

Groups *Presolve*

MSK_IPAR_PRESOLVE_LEVEL

Currently not used.

Default -1

Accepted [-inf; +inf]

Example `mosek -d MSK_IPAR_PRESOLVE_LEVEL -1 file`

Groups *Overall solver, Presolve*

MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH

Controls linear dependency check in presolve. The linear dependency check is potentially computationally expensive.

Default 100

Accepted [-inf; +inf]

Example `mosek -d MSK_IPAR_PRESOLVE_LINDEP_ABS_WORK_TRH 100 file`

Groups *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH

Controls linear dependency check in presolve. The linear dependency check is potentially computationally expensive.

Default 100

Accepted [-inf; +inf]

Example `mosek -d MSK_IPAR_PRESOLVE_LINDEP_REL_WORK_TRH 100 file`

Groups *Presolve*

MSK_IPAR_PRESOLVE_LINDEP_USE

Controls whether the linear constraints are checked for linear dependencies.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_PRESOLVE_LINDEP_USE MSK_ON file`

Groups *Presolve*

MSK_IPAR_PRESOLVE_MAX_NUM_PASS

Control the maximum number of times presolve passes over the problem. A negative value implies **MOSEK** decides.

Default -1

Accepted [-inf; +inf]

Example `mosek -d MSK_IPAR_PRESOLVE_MAX_NUM_PASS -1 file`

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]

Example `mosek -d MSK_IPAR_PRESOLVE_MAX_NUM_REDUCTIONS -1 file`

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*

Example `mosek -d MSK_IPAR_PRESOLVE_USE MSK_PRESOLVE_MODE_FREE file`

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*

Example `mosek -d MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER MSK_OPTIMIZER_FREE file`

Groups *Overall solver*

MSK_IPAR_PTF_WRITE_TRANSFORM

If *MSK_IPAR_PTF_WRITE_TRANSFORM* is *MSK_ON*, constraint blocks with identifiable conic slacks are transformed into conic constraints and the slacks are eliminated.

Default *ON*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_PTF_WRITE_TRANSFORM MSK_ON file`

Groups *Data input/output*

MSK_IPAR_READ_DEBUG

Turns on additional debugging information when reading files.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_READ_DEBUG MSK_OFF file`

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.

Example `mosek -d MSK_IPAR_READ_KEEP_FREE_CON MSK_OFF file`

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*

Example `mosek -d MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU MSK_OFF file`

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*

Example `mosek -d MSK_IPAR_READ_LP_QUOTED_NAMES MSK_ON file`

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*

Example `mosek -d MSK_IPAR_READ_MPS_FORMAT MSK_MPS_FORMAT_FREE file`

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]

Example `mosek -d MSK_IPAR_READ_MPS_WIDTH 1024 file`

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*

Example `mosek -d MSK_IPAR_READ_TASK_IGNORE_PARAM MSK_OFF file`

Groups *Data input/output*

MSK_IPAR_REMOVE_UNUSED_SOLUTIONS

Removes unused solutions before the optimization is performed.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_REMOVE_UNUSED_SOLUTIONS MSK_OFF file`

Groups *Overall system*

MSK_IPAR_SENSITIVITY_ALL

Not applicable.

Default *OFF*

Accepted *ON, OFF*

Example `mosek -d MSK_IPAR_SENSITIVITY_ALL MSK_OFF file`

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*

Example `mosek -d MSK_IPAR_SENSITIVITY_OPTIMIZER MSK_OPTIMIZER_FREE_SIMPLEX file`

Groups *Overall solver, Simplex optimizer*

MSK_IPAR_SENSITIVITY_TYPE

Controls which type of sensitivity analysis is to be performed.

Default *BASIS*

Accepted *BASIS*

Example `mosek -d MSK_IPAR_SENSITIVITY_TYPE MSK_SENSITIVITY_TYPE_BASIS file`

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*

Example `mosek -d MSK_IPAR_SIM_BASIS_FACTOR_USE MSK_ON file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_DEGEN

Controls how aggressively degeneration is handled.

Default *FREE*

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

Example `mosek -d MSK_IPAR_SIM_DEGEN MSK_SIM_DEGEN_FREE file`

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 $(100 - x) \bmod f_v$ entries, where f_v is the number of fixed variables.

Default 90

Accepted $[0; +\infty]$

Example `mosek -d MSK_IPAR_SIM_DUAL_CRASH 90 file`

Groups *Dual simplex*

MSK_IPAR_SIM_DUAL_PHASEONE_METHOD

An experimental feature.

Default 0

Accepted $[0; 10]$

Example `mosek -d MSK_IPAR_SIM_DUAL_PHASEONE_METHOD 0 file`

Groups *Simplex optimizer*

MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION

The dual simplex optimizer can use a so-called restricted selection/pricing strategy to choose 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 $[0; 100]$

Example `mosek -d MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION 50 file`

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*

Example `mosek -d MSK_IPAR_SIM_DUAL_SELECTION MSK_SIM_SELECTION_FREE
file`

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*

Example `mosek -d MSK_IPAR_SIM_EXPLOIT_DUPVEC MSK_SIM_EXPLOIT_DUPVEC_OFF file`

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*

Example `mosek -d MSK_IPAR_SIM_HOTSTART MSK_SIM_HOTSTART_FREE file`

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.

Example `mosek -d MSK_IPAR_SIM_HOTSTART_LU MSK_ON file`

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]

Example `mosek -d MSK_IPAR_SIM_MAX_ITERATIONS 10000000 file`

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]

Example `mosek -d MSK_IPAR_SIM_MAX_NUM_SETBACKS 250 file`

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*

Example `mosek -d MSK_IPAR_SIM_NON_SINGULAR MSK_ON file`

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]

Example mosek -d MSK_IPAR_SIM_PRIMAL_CRASH 90 file

Groups *Primal simplex*

MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD

An experimental feature.

Default 0

Accepted [0; 10]

Example mosek -d MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD 0 file

Groups *Simplex optimizer*

MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION

The primal simplex optimizer can use a so-called restricted selection/pricing strategy to choose 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]

Example mosek -d MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION 50 file

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*

Example mosek -d MSK_IPAR_SIM_PRIMAL_SELECTION MSK_SIM_SELECTION_FREE
file

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]

Example mosek -d MSK_IPAR_SIM_REFACTOR_FREQ 0 file

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*

Example mosek -d MSK_IPAR_SIM_REFORMULATION MSK_SIM_REFORMULATION_OFF
file

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*

Example mosek -d MSK_IPAR_SIM_SAVE_LU MSK_OFF file

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*

Example mosek -d MSK_IPAR_SIM_SCALING MSK_SCALING_FREE file

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*

Example mosek -d MSK_IPAR_SIM_SCALING_METHOD MSK_SCALING_METHOD_POW2
file

Groups *Simplex optimizer*

MSK_IPAR_SIM_SEED

Sets the random seed used for randomization in the simplex optimizers.

Default 23456

Accepted [0; 32749]

Example mosek -d MSK_IPAR_SIM_SEED 23456 file

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*

Example mosek -d MSK_IPAR_SIM_SOLVE_FORM MSK_SOLVE_FREE file

Groups *Simplex optimizer*

MSK_IPAR_SIM_STABILITY_PRIORITY

Controls how high priority the numerical stability should be given.

Default 50

Accepted [0; 100]

Example mosek -d MSK_IPAR_SIM_STABILITY_PRIORITY 50 file

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*

Example mosek -d MSK_IPAR_SIM_SWITCH_OPTIMIZER MSK_OFF file

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*

Example `mosek -d MSK_IPAR_SOL_FILTER_KEEP_BASIC MSK_OFF file`

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*

Example `mosek -d MSK_IPAR_SOL_FILTER_KEEP_RANGED MSK_OFF file`

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 must be specified. A negative value implies that no name contain blanks.

Default *-1*

Accepted *[-inf; +inf]*

Example `mosek -d MSK_IPAR_SOL_READ_NAME_WIDTH -1 file`

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]*

Example `mosek -d MSK_IPAR_SOL_READ_WIDTH 1024 file`

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*

Example `mosek -d MSK_IPAR_SOLUTION_CALLBACK MSK_OFF file`

Groups *Progress callback, Overall solver*

MSK_IPAR_TIMING_LEVEL

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

Default *1*

Accepted *[0; +inf]*

Example `mosek -d MSK_IPAR_TIMING_LEVEL 1 file`

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*

Example `mosek -d MSK_IPAR_WRITE_BAS_CONSTRAINTS MSK_ON file`

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*

Example mosek -d MSK_IPAR_WRITE_BAS_HEAD MSK_ON file

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*

Example mosek -d MSK_IPAR_WRITE_BAS_VARIABLES MSK_ON file

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

MSK_IPAR_WRITE_COMPRESSION

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

Default 9

Accepted [0; +inf]

Example mosek -d MSK_IPAR_WRITE_COMPRESSION 9 file

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*

Example mosek -d MSK_IPAR_WRITE_DATA_PARAM MSK_OFF file

Groups *Data input/output*

MSK_IPAR_WRITE_FREE_CON

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

Default *ON*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_FREE_CON MSK_ON file

Groups *Data input/output*

MSK_IPAR_WRITE_GENERIC_NAMES

Controls whether generic names should be used instead of user-defined names when writing to the data file.

Default *OFF*

Accepted *ON, OFF*

Example mosek -d MSK_IPAR_WRITE_GENERIC_NAMES MSK_OFF file

Groups *Data input/output*

MSK_IPAR_WRITE_GENERIC_NAMES_IO

Index origin used in generic names.

Default 1

Accepted [0; +inf]

Example mosek -d MSK_IPAR_WRITE_GENERIC_NAMES_IO 1 file

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 be written to the current output file format.

Example `mosek -d MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS MSK_OFF file`

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*

Example `mosek -d MSK_IPAR_WRITE_INT_CONSTRAINTS MSK_ON file`

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*

Example `mosek -d MSK_IPAR_WRITE_INT_HEAD MSK_ON file`

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*

Example `mosek -d MSK_IPAR_WRITE_INT_VARIABLES MSK_ON file`

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*

Example `mosek -d MSK_IPAR_WRITE_LP_FULL_OBJ MSK_ON file`

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]

Example `mosek -d MSK_IPAR_WRITE_LP_LINE_WIDTH 80 file`

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*

Example `mosek -d MSK_IPAR_WRITE_LP_QUOTED_NAMES MSK_ON file`

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*

Example `mosek -d MSK_IPAR_WRITE_LP_STRICT_FORMAT MSK_OFF file`

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]

Example mosek -d MSK_IPAR_WRITE_LP_TERMS_PER_LINE 10 file

Groups *Data input/output*

MSK_IPAR_WRITE_MPS_FORMAT

Controls in which format the MPS is written.

Default *FREE*

Accepted *STRICT, RELAXED, FREE, CPLEX*

Example mosek -d MSK_IPAR_WRITE_MPS_FORMAT MSK_MPS_FORMAT_FREE file

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, OFF*

Example mosek -d MSK_IPAR_WRITE_MPS_INT MSK_ON file

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]

Example mosek -d MSK_IPAR_WRITE_PRECISION 15 file

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*

Example mosek -d MSK_IPAR_WRITE_SOL_BARVARIABLES MSK_ON file

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*

Example mosek -d MSK_IPAR_WRITE_SOL_CONSTRAINTS MSK_ON file

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*

Example mosek -d MSK_IPAR_WRITE_SOL_HEAD MSK_ON file

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*
Example `mosek -d MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES MSK_OFF file`
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*
Example `mosek -d MSK_IPAR_WRITE_SOL_VARIABLES MSK_ON file`
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*
Example `mosek -d MSK_IPAR_WRITE_TASK_INC_SOL MSK_ON file`
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*
Example `mosek -d MSK_IPAR_WRITE_XML_MODE MSK_WRITE_XML_MODE_ROW file`
Groups *Data input/output*

9.3.3 String parameters

MSK_SPAR_BAS_SOL_FILE_NAME

Name of the bas solution file.

Accepted Any valid file name.
Example `mosek -d MSK_SPAR_BAS_SOL_FILE_NAME somevalue file`
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.
Example `mosek -d MSK_SPAR_DATA_FILE_NAME somevalue file`
Groups *Data input/output*

MSK_SPAR_DEBUG_FILE_NAME

MOSEK debug file.

Accepted Any valid file name.
Example `mosek -d MSK_SPAR_DEBUG_FILE_NAME somevalue file`
Groups *Data input/output*

MSK_SPAR_INT_SOL_FILE_NAME

Name of the int solution file.

Accepted Any valid file name.
Example `mosek -d MSK_SPAR_INT_SOL_FILE_NAME somevalue file`
Groups *Data input/output, Solution input/output*

MSK_SPAR_ITR_SOL_FILE_NAME

Name of the itr solution file.

Accepted Any valid file name.

Example mosek -d MSK_SPAR_ITR_SOL_FILE_NAME somevalue file

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

MSK_SPAR_MIO_DEBUG_STRING

For internal debugging purposes.

Accepted Any valid string.

Example mosek -d MSK_SPAR_MIO_DEBUG_STRING somevalue file

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.

Example mosek -d MSK_SPAR_PARAM_COMMENT_SIGN %% file

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.

Example mosek -d MSK_SPAR_PARAM_READ_FILE_NAME somevalue file

Groups *Data input/output*

MSK_SPAR_PARAM_WRITE_FILE_NAME

The parameter database is written to this file.

Accepted Any valid file name.

Example mosek -d MSK_SPAR_PARAM_WRITE_FILE_NAME somevalue file

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.

Example mosek -d MSK_SPAR_READ_MPS_BOU_NAME somevalue file

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.

Example mosek -d MSK_SPAR_READ_MPS_OBJ_NAME somevalue file

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.

Example mosek -d MSK_SPAR_READ_MPS_RAN_NAME somevalue file

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.

Example `mosek -d MSK_SPAR_READ_MPS_RHS_NAME somevalue file`

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.

Example `mosek -d MSK_SPAR_REMOTE_ACCESS_TOKEN somevalue file`

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.

Example `mosek -d MSK_SPAR_SENSITIVITY_FILE_NAME somevalue file`

Groups *Data input/output*

MSK_SPAR_SENSITIVITY_RES_FILE_NAME

Accepted Any valid string.

Example `mosek -d MSK_SPAR_SENSITIVITY_RES_FILE_NAME somevalue file`

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 $xc[i] \geq blc[i] + 0.5$ should be listed. An empty filter means that no filter is applied.

Accepted Any valid filter.

Example `mosek -d MSK_SPAR_SOL_FILTER_XC_LOW somevalue file`

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 $xc[i] \leq buc[i] - 0.5$ should be listed. An empty filter means that no filter is applied.

Accepted Any valid filter.

Example `mosek -d MSK_SPAR_SOL_FILTER_XC_UPR somevalue file`

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 $xx[j] \geq 0.5$ should be listed, whereas "+0.5" means that all constraints having $xx[j] \geq blx[j] + 0.5$ should be listed. An empty filter means no filter is applied.

Accepted Any valid filter.

Example `mosek -d MSK_SPAR_SOL_FILTER_XX_LOW somevalue file`

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 $xx[j] \leq bux[j] - 0.5$ should be listed. An empty filter means no filter is applied.

Accepted Any valid file name.

Example `mosek -d MSK_SPAR_SOL_FILTER_XX_UPR somevalue file`

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

MSK_SPAR_STAT_FILE_NAME

Statistics file name.

Accepted Any valid file name.

Example `mosek -d MSK_SPAR_STAT_FILE_NAME somevalue file`

Groups *Data input/output*

MSK_SPAR_STAT_KEY

Key used when writing the summary file.

Accepted Any valid string.

Example `mosek -d MSK_SPAR_STAT_KEY somevalue file`

Groups *Data input/output*

MSK_SPAR_STAT_NAME

Name used when writing the statistics file.

Accepted Any valid XML string.

Example `mosek -d MSK_SPAR_STAT_NAME somevalue file`

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.

Example `mosek -d MSK_SPAR_WRITE_LP_GEN_VAR_NAME xmskgen file`

Groups *Data input/output*

9.4 Response codes

Response codes include:

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

The numerical code (in brackets) identifies the response in error messages and in the log output.

9.4.1 Termination

MSK_RES_OK (0)

No error occurred.

MSK_RES_TRM_MAX_ITERATIONS (10000)

The optimizer terminated at the maximum number of iterations.

MSK_RES_TRM_MAX_TIME (10001)

The optimizer terminated at the maximum amount of time.

MSK_RES_TRM_OBJECTIVE_RANGE (10002)

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

MSK_RES_TRM_MIO_NUM_RELAXS (10008)

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

MSK_RES_TRM_MIO_NUM_BRANCHES (10009)

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

MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS (10015)

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

MSK_RES_TRM_STALL (10006)

The optimizer is terminated due to slow progress.

Stalling means that numerical problems prevent the optimizer from making reasonable progress and that it makes 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 feasible or 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 the solution. If the solution status is 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.

MSK_RES_TRM_USER_CALLBACK (10007)

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

MSK_RES_TRM_MAX_NUM_SETBACKS (10020)

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 (10025)

The optimizer terminated due to numerical problems.

MSK_RES_TRM_INTERNAL (10030)

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

MSK_RES_TRM_INTERNAL_STOP (10031)

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

9.4.2 Warnings

MSK_RES_WRN_OPEN_PARAM_FILE (50)

The parameter file could not be opened.

MSK_RES_WRN_LARGE_BOUND (51)

A numerically large bound value is specified.

MSK_RES_WRN_LARGE_LO_BOUND (52)

A numerically large lower bound value is specified.

MSK_RES_WRN_LARGE_UP_BOUND (53)

A numerically large upper bound value is specified.

MSK_RES_WRN_LARGE_CON_FX (54)

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

MSK_RES_WRN_LARGE_CJ (57)

A numerically large value is specified for one c_j .

MSK_RES_WRN_LARGE_AIJ (62)

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 (63)

One or more zero elements are specified in A .

MSK_RES_WRN_NAME_MAX_LEN (65)

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

MSK_RES_WRN_SPAR_MAX_LEN (66)

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

MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR (70)

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

MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR (71)

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

MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR (72)

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

MSK_RES_WRN_LP_OLD_QUAD_FORMAT (80)
 Missing $\sqrt{2}$ after quadratic expressions in bound or objective.

MSK_RES_WRN_LP_DROP_VARIABLE (85)
 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 (200)
 Non-zero elements specified in the upper triangle of a matrix were ignored.

MSK_RES_WRN_DROPPED_NZ_QOBJ (201)
 One or more non-zero elements were dropped in the Q matrix in the objective.

MSK_RES_WRN_IGNORE_INTEGER (250)
 Ignored integer constraints.

MSK_RES_WRN_NO_GLOBAL_OPTIMIZER (251)
 No global optimizer is available.

MSK_RES_WRN_MIO_INFEASIBLE_FINAL (270)
 The final mixed-integer problem with all the integer variables fixed at their optimal values is infeasible.

MSK_RES_WRN_SOL_FILTER (300)
 Invalid solution filter is specified.

MSK_RES_WRN_UNDEF_SOL_FILE_NAME (350)
 Undefined name occurred in a solution.

MSK_RES_WRN_SOL_FILE_IGNORED_CON (351)
 One or more lines in the constraint section were ignored when reading a solution file.

MSK_RES_WRN_SOL_FILE_IGNORED_VAR (352)
 One or more lines in the variable section were ignored when reading a solution file.

MSK_RES_WRN_TOO_FEW_BASIS_VARS (400)
 An incomplete basis has been specified. Too few basis variables are specified.

MSK_RES_WRN_TOO_MANY_BASIS_VARS (405)
 A basis with too many variables has been specified.

MSK_RES_WRN_LICENSE_EXPIRE (500)
 The license expires.

MSK_RES_WRN_LICENSE_SERVER (501)
 The license server is not responding.

MSK_RES_WRN_EMPTY_NAME (502)
 A variable or constraint name is empty. The output file may be invalid.

MSK_RES_WRN_USING_GENERIC_NAMES (503)
 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 (505)
 The license expires.

MSK_RES_WRN_PARAM_NAME_DOUB (510)
 The parameter name is not recognized as a double parameter.

MSK_RES_WRN_PARAM_NAME_INT (511)
 The parameter name is not recognized as an integer parameter.

MSK_RES_WRN_PARAM_NAME_STR (512)
 The parameter name is not recognized as a string parameter.

MSK_RES_WRN_PARAM_STR_VALUE (515)
 The string is not recognized as a symbolic value for the parameter.

MSK_RES_WRN_PARAM_IGNORED_CMIO (516)
 A parameter was ignored by the conic mixed integer optimizer.

MSK_RES_WRN_ZEROS_IN_SPARSE_ROW (705)
 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 (710)
 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 (800)
 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 (801)
The eliminator is skipped at least once due to lack of space.

MSK_RES_WRN_PRESOLVE_OUTOFSPACE (802)
The presolve is incomplete due to lack of space.

MSK_RES_WRN_WRITE_CHANGED_NAMES (803)
Some names were changed because they were invalid for the output file format.

MSK_RES_WRN_WRITE_DISCARDED_CFIX (804)
The fixed objective term could not be converted to a variable and was discarded in the output file.

MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES (850)
Two constraint names are identical.

MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES (851)
Two variable names are identical.

MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES (852)
Two barvariable names are identical.

MSK_RES_WRN_DUPLICATE_CONE_NAMES (853)
Two cone names are identical.

MSK_RES_WRN_ANA_LARGE_BOUNDS (900)
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 $+\text{inf}$ or $-\text{inf}$.

MSK_RES_WRN_ANA_C_ZERO (901)
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 (902)
This warning is issued by the problem analyzer, if columns, in which all coefficients are zero, are found.

MSK_RES_WRN_ANA_CLOSE_BOUNDS (903)
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 (904)
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 (930)
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 problem, or to fix all the variables in the cone to 0.

MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO (931)
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 problem, or to fix all the variables in the cone to 0.

MSK_RES_WRN_EXP_CONES_WITH_VARIABLES_FIXED_AT_ZERO (932)
For at least one exponential cone $x \geq y \exp(z/y)$ either the variable x or y 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 problem, or to fix all the variables in the cone to 0.

MSK_RES_WRN_POW_CONES_WITH_ROOT_FIXED_AT_ZERO (933)
For at least one power 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 problem, or to fix all the variables in the cone to 0.

MSK_RES_WRN_NO_DUALIZER (950)
No automatic dualizer is available for the specified problem. The primal problem is solved.

MSK_RES_WRN_SYM_MAT_LARGE (960)
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.

9.4.3 Errors

MSK_RES_ERR_LICENSE (1000)

Invalid license.

MSK_RES_ERR_LICENSE_EXPIRED (1001)

The license has expired.

MSK_RES_ERR_LICENSE_VERSION (1002)

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

MSK_RES_ERR_SIZE_LICENSE (1005)

The problem is bigger than the license.

MSK_RES_ERR_PROB_LICENSE (1006)

The software is not licensed to solve the problem.

MSK_RES_ERR_FILE_LICENSE (1007)

Invalid license file.

MSK_RES_ERR_MISSING_LICENSE_FILE (1008)

MOSEK cannot find license file or a token server. See the **MOSEK** licensing manual for details.

MSK_RES_ERR_SIZE_LICENSE_CON (1010)

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

MSK_RES_ERR_SIZE_LICENSE_VAR (1011)

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

MSK_RES_ERR_SIZE_LICENSE_INTVAR (1012)

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

MSK_RES_ERR_OPTIMIZER_LICENSE (1013)

The optimizer required is not licensed.

MSK_RES_ERR_FLEXLM (1014)

The FLEXlm license manager reported an error.

MSK_RES_ERR_LICENSE_SERVER (1015)

The license server is not responding.

MSK_RES_ERR_LICENSE_MAX (1016)

Maximum number of licenses is reached.

MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON (1017)

The MOSEKLM license manager daemon is not up and running.

MSK_RES_ERR_LICENSE_FEATURE (1018)

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 (1019)

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

MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE (1020)

The license system cannot allocate the memory required.

MSK_RES_ERR_LICENSE_CANNOT_CONNECT (1021)

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

MSK_RES_ERR_LICENSE_INVALID_HOSTID (1025)

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

MSK_RES_ERR_LICENSE_SERVER_VERSION (1026)

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

MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT (1027)

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 (1028)

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

MSK_RES_ERR_OLDER_DLL (1035)
The dynamic link library is older than the specified version.

MSK_RES_ERR_NEWER_DLL (1036)
The dynamic link library is newer than the specified version.

MSK_RES_ERR_LINK_FILE_DLL (1040)
A file cannot be linked to a stream in the DLL version.

MSK_RES_ERR_THREAD_MUTEX_INIT (1045)
Could not initialize a mutex.

MSK_RES_ERR_THREAD_MUTEX_LOCK (1046)
Could not lock a mutex.

MSK_RES_ERR_THREAD_MUTEX_UNLOCK (1047)
Could not unlock a mutex.

MSK_RES_ERR_THREAD_CREATE (1048)
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 (1049)
Could not initialize a condition.

MSK_RES_ERR_UNKNOWN (1050)
Unknown error.

MSK_RES_ERR_SPACE (1051)
Out of space.

MSK_RES_ERR_FILE_OPEN (1052)
Error while opening a file.

MSK_RES_ERR_FILE_READ (1053)
File read error.

MSK_RES_ERR_FILE_WRITE (1054)
File write error.

MSK_RES_ERR_DATA_FILE_EXT (1055)
The data file format cannot be determined from the file name.

MSK_RES_ERR_INVALID_FILE_NAME (1056)
An invalid file name has been specified.

MSK_RES_ERR_INVALID_SOL_FILE_NAME (1057)
An invalid file name has been specified.

MSK_RES_ERR_END_OF_FILE (1059)
End of file reached.

MSK_RES_ERR_NULL_ENV (1060)
`env` is a NULL pointer.

MSK_RES_ERR_NULL_TASK (1061)
`task` is a NULL pointer.

MSK_RES_ERR_INVALID_STREAM (1062)
An invalid stream is referenced.

MSK_RES_ERR_NO_INIT_ENV (1063)
`env` is not initialized.

MSK_RES_ERR_INVALID_TASK (1064)
The `task` is invalid.

MSK_RES_ERR_NULL_POINTER (1065)
An argument to a function is unexpectedly a NULL pointer.

MSK_RES_ERR_LIVING_TASKS (1066)
All tasks associated with an environment must be deleted before the environment is deleted. There are still some undeleted tasks.

MSK_RES_ERR_BLANK_NAME (1070)
An all blank name has been specified.

MSK_RES_ERR_DUP_NAME (1071)
The same name was used multiple times for the same problem item type.

MSK_RES_ERR_FORMAT_STRING (1072)
The name format string is invalid.

MSK_RES_ERR_INVALID_OBJ_NAME (1075)
 An invalid objective name is specified.

MSK_RES_ERR_INVALID_CON_NAME (1076)
 An invalid constraint name is used.

MSK_RES_ERR_INVALID_VAR_NAME (1077)
 An invalid variable name is used.

MSK_RES_ERR_INVALID_CONE_NAME (1078)
 An invalid cone name is used.

MSK_RES_ERR_INVALID_BARVAR_NAME (1079)
 An invalid symmetric matrix variable name is used.

MSK_RES_ERR_SPACE_LEAKING (1080)
MOSEK is leaking memory. This can be due to either an incorrect use of **MOSEK** or a bug.

MSK_RES_ERR_SPACE_NO_INFO (1081)
 No available information about the space usage.

MSK_RES_ERR_READ_FORMAT (1090)
 The specified format cannot be read.

MSK_RES_ERR_MPS_FILE (1100)
 An error occurred while reading an MPS file.

MSK_RES_ERR_MPS_INV_FIELD (1101)
 A field in the MPS file is invalid. Probably it is too wide.

MSK_RES_ERR_MPS_INV_MARKER (1102)
 An invalid marker has been specified in the MPS file.

MSK_RES_ERR_MPS_NULL_CON_NAME (1103)
 An empty constraint name is used in an MPS file.

MSK_RES_ERR_MPS_NULL_VAR_NAME (1104)
 An empty variable name is used in an MPS file.

MSK_RES_ERR_MPS_UNDEF_CON_NAME (1105)
 An undefined constraint name occurred in an MPS file.

MSK_RES_ERR_MPS_UNDEF_VAR_NAME (1106)
 An undefined variable name occurred in an MPS file.

MSK_RES_ERR_MPS_INV_CON_KEY (1107)
 An invalid constraint key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_BOUND_KEY (1108)
 An invalid bound key occurred in an MPS file.

MSK_RES_ERR_MPS_INV_SEC_NAME (1109)
 An invalid section name occurred in an MPS file.

MSK_RES_ERR_MPS_NO_OBJECTIVE (1110)
 No objective is defined in an MPS file.

MSK_RES_ERR_MPS_SPLITTED_VAR (1111)
 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 (1112)
 A constraint name was specified multiple times in the ROWS section.

MSK_RES_ERR_MPS_MUL_QSEC (1113)
 Multiple QSECTIONs are specified for a constraint in the MPS data file.

MSK_RES_ERR_MPS_MUL_QOBJ (1114)
 The Q term in the objective is specified multiple times in the MPS data file.

MSK_RES_ERR_MPS_INV_SEC_ORDER (1115)
 The sections in the MPS data file are not in the correct order.

MSK_RES_ERR_MPS_MUL_CSEC (1116)
 Multiple CSECTIONs are given the same name.

MSK_RES_ERR_MPS_CONE_TYPE (1117)
 Invalid cone type specified in a CSECTION.

MSK_RES_ERR_MPS_CONE_OVERLAP (1118)
 A variable is specified to be a member of several cones.

MSK_RES_ERR_MPS_CONE_REPEAT (1119)
 A variable is repeated within the CSECTION.

MSK_RES_ERR_MPS_NON_SYMMETRIC_Q (1120)
 A non symmetric matrix has been specified.

MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT (1121)
 Duplicate elements is specified in a Q matrix.

MSK_RES_ERR_MPS_INVALID_OBJSENSE (1122)
 An invalid objective sense is specified.

MSK_RES_ERR_MPS_TAB_IN_FIELD2 (1125)
 A tab char occurred in field 2.

MSK_RES_ERR_MPS_TAB_IN_FIELD3 (1126)
 A tab char occurred in field 3.

MSK_RES_ERR_MPS_TAB_IN_FIELD5 (1127)
 A tab char occurred in field 5.

MSK_RES_ERR_MPS_INVALID_OBJ_NAME (1128)
 An invalid objective name is specified.

MSK_RES_ERR_LP_INCOMPATIBLE (1150)
 The problem cannot be written to an LP formatted file.

MSK_RES_ERR_LP_EMPTY (1151)
 The problem cannot be written to an LP formatted file.

MSK_RES_ERR_LP_DUP_SLACK_NAME (1152)
 The name of the slack variable added to a ranged constraint already exists.

MSK_RES_ERR_WRITE_MPS_INVALID_NAME (1153)
 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 (1154)
 A variable name is invalid when used in an LP formatted file.

MSK_RES_ERR_LP_FREE_CONSTRAINT (1155)
 Free constraints cannot be written in LP file format.

MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME (1156)
 Empty variable names cannot be written to OPF files.

MSK_RES_ERR_LP_FILE_FORMAT (1157)
 Syntax error in an LP file.

MSK_RES_ERR_WRITE_LP_FORMAT (1158)
 Problem cannot be written as an LP file.

MSK_RES_ERR_READ_LP_MISSING_END_TAG (1159)
 Syntax error in LP file. Possibly missing End tag.

MSK_RES_ERR_LP_FORMAT (1160)
 Syntax error in an LP file.

MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME (1161)
 An auto-generated name is not unique.

MSK_RES_ERR_READ_LP_NONEXISTING_NAME (1162)
 A variable never occurred in objective or constraints.

MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM (1163)
 The problem contains cones that cannot be written to an LP formatted file.

MSK_RES_ERR_LP_WRITE_GECO_PROBLEM (1164)
 The problem contains general convex terms that cannot be written to an LP formatted file.

MSK_RES_ERR_WRITING_FILE (1166)
 An error occurred while writing file

MSK_RES_ERR_PTF_FORMAT (1167)
 Syntax error in an PTF file

MSK_RES_ERR_OPF_FORMAT (1168)
 Syntax error in an OPF file

MSK_RES_ERR_OPF_NEW_VARIABLE (1169)
 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 (1170)
 An invalid name occurred in a solution file.

MSK_RES_ERR_LP_INVALID_CON_NAME (1171)
 A constraint name is invalid when used in an LP formatted file.

MSK_RES_ERR_OPF_PREMATURE_EOF (1172)
Premature end of file in an OPF file.

MSK_RES_ERR_JSON_SYNTAX (1175)
Syntax error in an JSON data

MSK_RES_ERR_JSON_STRING (1176)
Error in JSON string.

MSK_RES_ERR_JSON_NUMBER_OVERFLOW (1177)
Invalid number entry - wrong type or value overflow.

MSK_RES_ERR_JSON_FORMAT (1178)
Error in an JSON Task file

MSK_RES_ERR_JSON_DATA (1179)
Inconsistent data in JSON Task file

MSK_RES_ERR_JSON_MISSING_DATA (1180)
Missing data section in JSON task file.

MSK_RES_ERR_ARGUMENT_LENNEQ (1197)
Incorrect length of arguments.

MSK_RES_ERR_ARGUMENT_TYPE (1198)
Incorrect argument type.

MSK_RES_ERR_NUM_ARGUMENTS (1199)
Incorrect number of function arguments.

MSK_RES_ERR_IN_ARGUMENT (1200)
A function argument is incorrect.

MSK_RES_ERR_ARGUMENT_DIMENSION (1201)
A function argument is of incorrect dimension.

MSK_RES_ERR_SHAPE_IS_TOO_LARGE (1202)
The size of the n-dimensional shape is too large.

MSK_RES_ERR_INDEX_IS_TOO_SMALL (1203)
An index in an argument is too small.

MSK_RES_ERR_INDEX_IS_TOO_LARGE (1204)
An index in an argument is too large.

MSK_RES_ERR_PARAM_NAME (1205)
The parameter name is not correct.

MSK_RES_ERR_PARAM_NAME_DOUB (1206)
The parameter name is not correct for a double parameter.

MSK_RES_ERR_PARAM_NAME_INT (1207)
The parameter name is not correct for an integer parameter.

MSK_RES_ERR_PARAM_NAME_STR (1208)
The parameter name is not correct for a string parameter.

MSK_RES_ERR_PARAM_INDEX (1210)
Parameter index is out of range.

MSK_RES_ERR_PARAM_IS_TOO_LARGE (1215)
The parameter value is too large.

MSK_RES_ERR_PARAM_IS_TOO_SMALL (1216)
The parameter value is too small.

MSK_RES_ERR_PARAM_VALUE_STR (1217)
The parameter value string is incorrect.

MSK_RES_ERR_PARAM_TYPE (1218)
The parameter type is invalid.

MSK_RES_ERR_INF_DOUB_INDEX (1219)
A double information index is out of range for the specified type.

MSK_RES_ERR_INF_INT_INDEX (1220)
An integer information index is out of range for the specified type.

MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL (1221)
An index in an array argument is too small.

MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE (1222)
An index in an array argument is too large.

MSK_RES_ERR_INF_LINT_INDEX (1225)
A long integer information index is out of range for the specified type.

MSK_RES_ERR_ARG_IS_TOO_SMALL (1226)
The value of a argument is too small.

MSK_RES_ERR_ARG_IS_TOO_LARGE (1227)
The value of a argument is too large.

MSK_RES_ERR_INVALID_WHICHSOL (1228)
whichsol is invalid.

MSK_RES_ERR_INF_DOU_NAME (1230)
A double information name is invalid.

MSK_RES_ERR_INF_INT_NAME (1231)
An integer information name is invalid.

MSK_RES_ERR_INF_TYPE (1232)
The information type is invalid.

MSK_RES_ERR_INF_LINT_NAME (1234)
A long integer information name is invalid.

MSK_RES_ERR_INDEX (1235)
An index is out of range.

MSK_RES_ERR_WHICHSOL (1236)
The solution defined by *whichsol* does not exists.

MSK_RES_ERR_SOLITEM (1237)
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 (1238)
whichitem is unacceptable.

MSK_RES_ERR_MAXNUMCON (1240)
The maximum number of constraints specified is smaller than the number of constraints in the task.

MSK_RES_ERR_MAXNUMVAR (1241)
The maximum number of variables specified is smaller than the number of variables in the task.

MSK_RES_ERR_MAXNUMBARVAR (1242)
The maximum number of semidefinite variables specified is smaller than the number of semidefinite variables in the task.

MSK_RES_ERR_MAXNUMQNZ (1243)
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 (1245)
The maximum number of non-zeros specified is too small.

MSK_RES_ERR_INVALID_IDX (1246)
A specified index is invalid.

MSK_RES_ERR_INVALID_MAX_NUM (1247)
A specified index is invalid.

MSK_RES_ERR_NUMCONLIM (1250)
Maximum number of constraints limit is exceeded.

MSK_RES_ERR_NUMVARLIM (1251)
Maximum number of variables limit is exceeded.

MSK_RES_ERR_TOO_SMALL_MAXNUMANZ (1252)
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 (1253)
aptre[j] is strictly smaller than *aptrb[j]* for some j .

MSK_RES_ERR_MUL_A_ELEMENT (1254)
An element in A is defined multiple times.

MSK_RES_ERR_INV_BK (1255)
Invalid bound key.

MSK_RES_ERR_INV_BKC (1256)
Invalid bound key is specified for a constraint.

MSK_RES_ERR_INV_BKX (1257)
An invalid bound key is specified for a variable.

MSK_RES_ERR_INV_VAR_TYPE (1258)

An invalid variable type is specified for a variable.

MSK_RES_ERR_SOLVER_PROBTYPE (1259)

Problem type does not match the chosen optimizer.

MSK_RES_ERR_OBJECTIVE_RANGE (1260)

Empty objective range.

MSK_RES_ERR_UNDEF_SOLUTION (1265)

MOSEK has the following solution types:

- an interior-point solution,
- a 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 (1266)

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

MSK_RES_ERR_INV_SKC (1267)

Invalid value in `skc`.

MSK_RES_ERR_INV_SKX (1268)

Invalid value in `skx`.

MSK_RES_ERR_INV_SKN (1274)

Invalid value in `skn`.

MSK_RES_ERR_INV_SK_STR (1269)

Invalid status key string encountered.

MSK_RES_ERR_INV_SK (1270)

Invalid status key code.

MSK_RES_ERR_INV_CONE_TYPE_STR (1271)

Invalid cone type string encountered.

MSK_RES_ERR_INV_CONE_TYPE (1272)

Invalid cone type code is encountered.

MSK_RES_ERR_INVALID_SURPLUS (1275)

Invalid surplus.

MSK_RES_ERR_INV_NAME_ITEM (1280)

An invalid name item code is used.

MSK_RES_ERR_PRO_ITEM (1281)

An invalid problem is used.

MSK_RES_ERR_INVALID_FORMAT_TYPE (1283)

Invalid format type.

MSK_RES_ERR_FIRSTI (1285)

Invalid `firsti`.

MSK_RES_ERR_LASTI (1286)

Invalid `lasti`.

MSK_RES_ERR_FIRSTJ (1287)

Invalid `firstj`.

MSK_RES_ERR_LASTJ (1288)

Invalid `lastj`.

MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL (1289)

A maximum length that is too small has been specified.

MSK_RES_ERR_NONLINEAR_EQUALITY (1290)

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

MSK_RES_ERR_NONCONVEX (1291)

The optimization problem is nonconvex.

MSK_RES_ERR_NONLINEAR_RANGED (1292)

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

MSK_RES_ERR_CON_Q_NOT_PSD (1293)

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 (1294)

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 (1295)

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 (1296)

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 (1299)

An invalid permutation array is specified.

MSK_RES_ERR_CONE_INDEX (1300)

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

MSK_RES_ERR_CONE_SIZE (1301)

A cone with incorrect number of members is specified.

MSK_RES_ERR_CONE_OVERLAP (1302)

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 (1303)

A variable is included multiple times in the cone.

MSK_RES_ERR_MAXNUMCONE (1304)

The value specified for `maxnumcone` is too small.

MSK_RES_ERR_CONE_TYPE (1305)

Invalid cone type specified.

MSK_RES_ERR_CONE_TYPE_STR (1306)

Invalid cone type specified.

MSK_RES_ERR_CONE_OVERLAP_APPEND (1307)

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

MSK_RES_ERR_REMOVE_CONE_VARIABLE (1310)

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

MSK_RES_ERR_APPENDING_TOO_BIG_CONE (1311)

Trying to append a too big cone.

MSK_RES_ERR_CONE_PARAMETER (1320)

An invalid cone parameter.

MSK_RES_ERR_SOL_FILE_INVALID_NUMBER (1350)

An invalid number is specified in a solution file.

MSK_RES_ERR_HUGE_C (1375)

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

MSK_RES_ERR_HUGE_AIJ (1380)

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 (1385)

An element in the A matrix is specified twice.

MSK_RES_ERR_LOWER_BOUND_IS_A_NAN (1390)

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

MSK_RES_ERR_UPPER_BOUND_IS_A_NAN (1391)

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

MSK_RES_ERR_INFINITY_BOUND (1400)
 A numerically huge bound value is specified.

MSK_RES_ERR_INV_QOBJ_SUBI (1401)
 Invalid value in `qosubi`.

MSK_RES_ERR_INV_QOBJ_SUBJ (1402)
 Invalid value in `qosubj`.

MSK_RES_ERR_INV_QOBJ_VAL (1403)
 Invalid value in `qoval`.

MSK_RES_ERR_INV_QCON_SUBK (1404)
 Invalid value in `qcsubk`.

MSK_RES_ERR_INV_QCON_SUBI (1405)
 Invalid value in `qcsubi`.

MSK_RES_ERR_INV_QCON_SUBJ (1406)
 Invalid value in `qcsubj`.

MSK_RES_ERR_INV_QCON_VAL (1407)
 Invalid value in `qcval`.

MSK_RES_ERR_QCON_SUBI_TOO_SMALL (1408)
 Invalid value in `qcsubi`.

MSK_RES_ERR_QCON_SUBI_TOO_LARGE (1409)
 Invalid value in `qcsubi`.

MSK_RES_ERR_QOBJ_UPPER_TRIANGLE (1415)
 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 (1417)
 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 (1420)
 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_TOO_SMALL_A_TRUNCATION_VALUE (1421)
 A too small value for the A truncation value is specified.

MSK_RES_ERR_INVALID_OBJECTIVE_SENSE (1445)
 An invalid objective sense is specified.

MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE (1446)
 The objective sense has not been specified before the optimization.

MSK_RES_ERR_Y_IS_UNDEFINED (1449)
 The solution item y is undefined.

MSK_RES_ERR_NAN_IN_DOUBLE_DATA (1450)
 An invalid floating point value was used in some double data.

MSK_RES_ERR_NAN_IN_BLC (1461)
 l^c contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_BUC (1462)
 u^c contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_C (1470)
 c contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_BLX (1471)
 l^x contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_NAN_IN_BUX (1472)
 u^x contains an invalid floating point value, i.e. a NaN.

MSK_RES_ERR_INVALID_AIJ (1473)
 $a_{i,j}$ contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_INVALID_CJ (1474)
 c_j contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_SYM_MAT_INVALID (1480)
 A symmetric matrix contains an invalid floating point value, i.e. a NaN or an infinite value.

MSK_RES_ERR_SYM_MAT_HUGE (1482)
 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 (1500)
Invalid problem type. Probably a nonconvex problem has been specified.

MSK_RES_ERR_MIXED_CONIC_AND_NL (1501)
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 (1503)
The global optimizer can only be applied to problems without semidefinite variables.

MSK_RES_ERR_INV_OPTIMIZER (1550)
An invalid optimizer has been chosen for the problem.

MSK_RES_ERR_MIO_NO_OPTIMIZER (1551)
No optimizer is available for the current class of integer optimization problems.

MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE (1552)
No optimizer is available for this class of optimization problems.

MSK_RES_ERR_FINAL_SOLUTION (1560)
An error occurred during the solution finalization.

MSK_RES_ERR_FIRST (1570)
Invalid `first`.

MSK_RES_ERR_LAST (1571)
Invalid index `last`. A given index was out of expected range.

MSK_RES_ERR_SLICE_SIZE (1572)
Invalid slice size specified.

MSK_RES_ERR_NEGATIVE_SURPLUS (1573)
Negative surplus.

MSK_RES_ERR_NEGATIVE_APPEND (1578)
Cannot append a negative number.

MSK_RES_ERR_POSTSOLVE (1580)
An error occurred during the postsolve. Please contact **MOSEK** support.

MSK_RES_ERR_OVERFLOW (1590)
A computation produced an overflow i.e. a very large number.

MSK_RES_ERR_NO_BASIS_SOL (1600)
No basic solution is defined.

MSK_RES_ERR_BASIS_FACTOR (1610)
The factorization of the basis is invalid.

MSK_RES_ERR_BASIS_SINGULAR (1615)
The basis is singular and hence cannot be factored.

MSK_RES_ERR_FACTOR (1650)
An error occurred while factorizing a matrix.

MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX (1700)
An optimization problem cannot be relaxed.

MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED (1701)
The relaxed problem could not be solved to optimality. Please consult the log file for further details.

MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND (1702)
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 (1710)
The feasibility repair does not support the specified problem type.

MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED (1711)
Computation the optimal relaxation failed. The cause may have been numerical problems.

MSK_RES_ERR_NAME_MAX_LEN (1750)
A name is longer than the buffer that is supposed to hold it.

MSK_RES_ERR_NAME_IS_NULL (1760)
The name buffer is a NULL pointer.

MSK_RES_ERR_INVALID_COMPRESSION (1800)
Invalid compression type.

MSK_RES_ERR_INVALID_IOMODE (1801)
Invalid io mode.

MSK_RES_ERR_NO_PRIMAL_INFEAS_CER (2000)
A certificate of primal infeasibility is not available.

MSK_RES_ERR_NO_DUAL_INFEAS_CER (2001)
 A certificate of infeasibility is not available.

MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK (2500)
 The required solution is not available.

MSK_RES_ERR_INV_MARKI (2501)
 Invalid value in marki.

MSK_RES_ERR_INV_MARKJ (2502)
 Invalid value in markj.

MSK_RES_ERR_INV_NUMI (2503)
 Invalid numi.

MSK_RES_ERR_INV_NUMJ (2504)
 Invalid numj.

MSK_RES_ERR_TASK_INCOMPATIBLE (2560)
 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 (2561)
 The Task file is invalid.

MSK_RES_ERR_TASK_WRITE (2562)
 Failed to write the task file.

MSK_RES_ERR_LU_MAX_NUM_TRIES (2800)
 Could not compute the LU factors of the matrix within the maximum number of allowed tries.

MSK_RES_ERR_INVALID_UTF8 (2900)
 An invalid UTF8 string is encountered.

MSK_RES_ERR_INVALID_WCHAR (2901)
 An invalid wchar string is encountered.

MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL (2950)
 No dual information is available for the integer solution.

MSK_RES_ERR_NO_SNX_FOR_BAS_SOL (2953)
 s_n^x is not available for the basis solution.

MSK_RES_ERR_INTERNAL (3000)
 An internal error occurred. Please report this problem.

MSK_RES_ERR_API_ARRAY_TOO_SMALL (3001)
 An input array was too short.

MSK_RES_ERR_API_CB_CONNECT (3002)
 Failed to connect a callback object.

MSK_RES_ERR_API_FATAL_ERROR (3005)
 An internal error occurred in the API. Please report this problem.

MSK_RES_ERR_API_INTERNAL (3999)
 An internal fatal error occurred in an interface function.

MSK_RES_ERR_SEN_FORMAT (3050)
 Syntax error in sensitivity analysis file.

MSK_RES_ERR_SEN_UNDEF_NAME (3051)
 An undefined name was encountered in the sensitivity analysis file.

MSK_RES_ERR_SEN_INDEX_RANGE (3052)
 Index out of range in the sensitivity analysis file.

MSK_RES_ERR_SEN_BOUND_INVALID_UP (3053)
 Analysis of upper bound requested for an index, where no upper bound exists.

MSK_RES_ERR_SEN_BOUND_INVALID_LO (3054)
 Analysis of lower bound requested for an index, where no lower bound exists.

MSK_RES_ERR_SEN_INDEX_INVALID (3055)
 Invalid range given in the sensitivity file.

MSK_RES_ERR_SEN_INVALID_REGEXP (3056)
 Syntax error in regexp or regexp longer than 1024.

MSK_RES_ERR_SEN_SOLUTION_STATUS (3057)
 No optimal solution found to the original problem given for sensitivity analysis.

MSK_RES_ERR_SEN_NUMERICAL (3058)
 Numerical difficulties encountered performing the sensitivity analysis.

MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE (3080)
Sensitivity analysis cannot be performed for the specified problem. Sensitivity analysis is only possible for linear problems.

MSK_RES_ERR_UNB_STEP_SIZE (3100)
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 (3101)
Some tasks related to this function call were identical. Unique tasks were expected.

MSK_RES_ERR_AD_INVALID_CODELIST (3102)
The code list data was invalid.

MSK_RES_ERR_INTERNAL_TEST_FAILED (3500)
An internal unit test function failed.

MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE (3600)
The problem type is not supported by the XML format.

MSK_RES_ERR_INVALID_AMPL_STUB (3700)
Invalid AMPL stub.

MSK_RES_ERR_INT64_TO_INT32_CAST (3800)
A 64 bit integer could not be cast to a 32 bit integer.

MSK_RES_ERR_SIZE_LICENSE_NUMCORES (3900)
The computer contains more cpu cores than the license allows for.

MSK_RES_ERR_INFEAS_UNDEFINED (3910)
The requested value is not defined for this solution type.

MSK_RES_ERR_NO_BARX_FOR_SOLUTION (3915)
There is no \bar{X} available for the solution specified. In particular note there are no \bar{X} defined for the basic and integer solutions.

MSK_RES_ERR_NO_BARS_FOR_SOLUTION (3916)
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 (3920)
The dimension of a symmetric matrix variable has to be greater than 0.

MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX (3940)
A row index specified for sparse symmetric matrix is invalid.

MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX (3941)
A column index specified for sparse symmetric matrix is invalid.

MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR (3942)
Only the lower triangular part of sparse symmetric matrix should be specified.

MSK_RES_ERR_SYM_MAT_INVALID_VALUE (3943)
The numerical value specified in a sparse symmetric matrix is not a floating point value.

MSK_RES_ERR_SYM_MAT_DUPLICATE (3944)
A value in a symmetric matrix has been specified more than once.

MSK_RES_ERR_INVALID_SYM_MAT_DIM (3950)
A sparse symmetric matrix of invalid dimension is specified.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_SYM_MAT (4000)
The file format does not support a problem with symmetric matrix variables.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CFIX (4001)
The file format does not support a problem with nonzero fixed term in c.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_RANGED_CONSTRAINTS (4002)
The file format does not support a problem with ranged constraints.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_FREE_CONSTRAINTS (4003)
The file format does not support a problem with free constraints.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CONES (4005)
The file format does not support a problem with conic constraints.

MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_NONLINEAR (4010)
The file format does not support a problem with nonlinear terms.

MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES (4500)
Two constraint names are identical.

MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES (4501)
Two variable names are identical.

MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES (4502)
Two barvariable names are identical.

MSK_RES_ERR_DUPLICATE_CONE_NAMES (4503)
Two cone names are identical.

MSK_RES_ERR_NON_UNIQUE_ARRAY (5000)
An array does not contain unique elements.

MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE (5005)
The value of a function argument is too large.

MSK_RES_ERR_MIO_INTERNAL (5010)
A fatal error occurred in the mixed integer optimizer. Please contact **MOSEK** support.

MSK_RES_ERR_INVALID_PROBLEM_TYPE (6000)
An invalid problem type.

MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS (6010)
Unhandled solution status.

MSK_RES_ERR_UPPER_TRIANGLE (6020)
An element in the upper triangle of a lower triangular matrix is specified.

MSK_RES_ERR_LAU_SINGULAR_MATRIX (7000)
A matrix is singular.

MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE (7001)
A matrix is not positive definite.

MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX (7002)
An invalid lower triangular matrix.

MSK_RES_ERR_LAU_UNKNOWN (7005)
An unknown error.

MSK_RES_ERR_LAU_ARG_M (7010)
Invalid argument m.

MSK_RES_ERR_LAU_ARG_N (7011)
Invalid argument n.

MSK_RES_ERR_LAU_ARG_K (7012)
Invalid argument k.

MSK_RES_ERR_LAU_ARG_TRANSA (7015)
Invalid argument transa.

MSK_RES_ERR_LAU_ARG_TRANSB (7016)
Invalid argument transb.

MSK_RES_ERR_LAU_ARG_UPLO (7017)
Invalid argument uplo.

MSK_RES_ERR_LAU_ARG_TRANS (7018)
Invalid argument trans.

MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX (7019)
An invalid sparse symmetric matrix is specified. Note only the lower triangular part with no duplicates is specified.

MSK_RES_ERR_CBF_PARSE (7100)
An error occurred while parsing an CBF file.

MSK_RES_ERR_CBF_OBJ_SENSE (7101)
An invalid objective sense is specified.

MSK_RES_ERR_CBF_NO_VARIABLES (7102)
No variables are specified.

MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS (7103)
Too many constraints specified.

MSK_RES_ERR_CBF_TOO_MANY_VARIABLES (7104)
Too many variables specified.

MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED (7105)
No version specified.

MSK_RES_ERR_CBF_SYNTAX (7106)
Invalid syntax.

MSK_RES_ERR_CBF_DUPLICATE_OBJ (7107)
Duplicate OBJ keyword.

MSK_RES_ERR_CBF_DUPLICATE_CON (7108)
Duplicate CON keyword.

MSK_RES_ERR_CBF_DUPLICATE_VAR (7109)
Duplicate VAR keyword.

MSK_RES_ERR_CBF_DUPLICATE_INT (7110)
Duplicate INT keyword.

MSK_RES_ERR_CBF_INVALID_VAR_TYPE (7111)
Invalid variable type.

MSK_RES_ERR_CBF_INVALID_CON_TYPE (7112)
Invalid constraint type.

MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION (7113)
Invalid domain dimension.

MSK_RES_ERR_CBF_DUPLICATE_OBJCOORD (7114)
Duplicate index in OBJCOORD.

MSK_RES_ERR_CBF_DUPLICATE_BCOORD (7115)
Duplicate index in BCOORD.

MSK_RES_ERR_CBF_DUPLICATE_ACOORD (7116)
Duplicate index in ACOORD.

MSK_RES_ERR_CBF_TOO_FEW_VARIABLES (7117)
Too few variables defined.

MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS (7118)
Too few constraints defined.

MSK_RES_ERR_CBF_TOO_FEW_INTS (7119)
Too few ints are specified.

MSK_RES_ERR_CBF_TOO_MANY_INTS (7120)
Too many ints are specified.

MSK_RES_ERR_CBF_INVALID_INT_INDEX (7121)
Invalid INT index.

MSK_RES_ERR_CBF_UNSUPPORTED (7122)
Unsupported feature is present.

MSK_RES_ERR_CBF_DUPLICATE_PSDVAR (7123)
Duplicate PSDVAR keyword.

MSK_RES_ERR_CBF_INVALID_PSDVAR_DIMENSION (7124)
Invalid PSDVAR dimension.

MSK_RES_ERR_CBF_TOO_FEW_PSDVAR (7125)
Too few variables defined.

MSK_RES_ERR_CBF_INVALID_EXP_DIMENSION (7126)
Invalid dimension of a exponential cone.

MSK_RES_ERR_CBF_DUPLICATE_POW_CONES (7130)
Multiple POWCONES specified.

MSK_RES_ERR_CBF_DUPLICATE_POW_STAR_CONES (7131)
Multiple POW*CONES specified.

MSK_RES_ERR_CBF_INVALID_POWER (7132)
Invalid power specified.

MSK_RES_ERR_CBF_POWER_CONE_IS_TOO_LONG (7133)
Power cone is too long.

MSK_RES_ERR_CBF_INVALID_POWER_CONE_INDEX (7134)
Invalid power cone index.

MSK_RES_ERR_CBF_INVALID_POWER_STAR_CONE_INDEX (7135)
Invalid power star cone index.

MSK_RES_ERR_CBF_UNHANDLED_POWER_CONE_TYPE (7136)
An unhandled power cone type.

MSK_RES_ERR_CBF_UNHANDLED_POWER_STAR_CONE_TYPE (7137)
An unhandled power star cone type.

MSK_RES_ERR_CBF_POWER_CONE_MISMATCH (7138)
The power cone does not match with it definition.

MSK_RES_ERR_CBF_POWER_STAR_CONE_MISMATCH (7139)
The power star cone does not match with its definition.

MSK_RES_ERR_CBF_INVALID_NUMBER_OF_CONES (7740)
Invalid number of cones.

MSK_RES_ERR_CBF_INVALID_DIMENSION_OF_CONES (7741)
Invalid dimension of cones.

MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER (7700)
An invalid root optimizer was selected for the problem type.

MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER (7701)
An invalid node optimizer was selected for the problem type.

MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD (7800)
The matrix defining the quadratic part of constraint is not positive semidefinite.

MSK_RES_ERR_TOCONIC_CONSTRAINT_FX (7801)
The quadratic constraint is an equality, thus not convex.

MSK_RES_ERR_TOCONIC_CONSTRAINT_RA (7802)
The quadratic constraint has finite lower and upper bound, and therefore it is not convex.

MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC (7803)
The constraint is not conic representable.

MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD (7804)
The matrix defining the quadratic part of the objective function is not positive semidefinite.

MSK_RES_ERR_SERVER_CONNECT (8000)
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 (8001)
Unexpected message or data from solver server.

MSK_RES_ERR_SERVER_STATUS (8002)
Server returned non-ok HTTP status code

MSK_RES_ERR_SERVER_TOKEN (8003)
The job ID specified is incorrect or invalid

MSK_RES_ERR_SERVER_PROBLEM_SIZE (8008)
The size of the problem exceeds the dimensions permitted by the instance of the OptServer where it was run.

9.5 Constants

9.5.1 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_RESERVED
Not currently in use.

9.5.2 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 a 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.

9.5.3 Mark

MSK_MARK_LO

The lower bound is selected for sensitivity analysis.

MSK_MARK_UP

The upper bound is selected for sensitivity analysis.

9.5.4 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.

9.5.5 Transposed matrix.

MSK_TRANSPOSE_NO

No transpose is applied.

MSK_TRANSPOSE_YES

A transpose is applied.

9.5.6 Triangular part of a symmetric matrix.

MSK_UPLO_LO

Lower part.

MSK_UPLO_UP

Upper part.

9.5.7 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.

9.5.8 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.

9.5.9 Hot-start type employed by the simplex optimizer

MSK_SIM_HOTSTART_NONE

The simplex optimizer performs a coldstart.

MSK_SIM_HOTSTART_FREE

The simplex optimizer 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.

9.5.10 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.

9.5.11 Solution purification employed optimizer.

MSK_PURIFY_NONE

The optimizer performs no solution purification.

MSK_PURIFY_PRIMAL

The optimizer purifies the primal solution.

MSK_PURIFY_DUAL

The optimizer purifies the dual solution.

MSK_PURIFY_PRIMAL_DUAL

The optimizer purifies both the primal and dual solution.

MSK_PURIFY_AUTO

TBD

9.5.12 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_BEGIN_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 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.

9.5.13 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.

9.5.14 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.

MSK_COMPRESS_ZSTD

The type of compression used is zstd compatible.

9.5.15 Cone types

MSK_CT_QUAD

The cone is a quadratic cone.

MSK_CT_RQUAD

The cone is a rotated quadratic cone.

MSK_CT_PEXP

A primal exponential cone.

MSK_CT_DEXP

A dual exponential cone.

MSK_CT_PPOW

A primal power cone.

MSK_CT_DPOW

A dual power cone.

MSK_CT_ZERO

The zero cone.

9.5.16 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.

9.5.17 SCopt operator types

MSK_OPR_ENT

Entropy

MSK_OPR_EXP

Exponential

MSK_OPR_LOG

Logarithm

MSK_OPR_POW

Power

MSK_OPR_SQRT

Square root

9.5.18 Cone types

MSK_SYMMAT_TYPE_SPARSE

Sparse symmetric matrix.

9.5.19 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_FREE_MPS

The data a free MPS formatted file.

MSK_DATA_FORMAT_TASK

Generic task dump file.

MSK_DATA_FORMAT_PTF
(P)retty (T)ext (F)ormat.
MSK_DATA_FORMAT_CB
Conic benchmark format,
MSK_DATA_FORMAT_JSON_TASK
JSON based task format.

9.5.20 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_CLIQUÉ_SEPARATION_TIME
Separation time for clique cuts.

MSK_DINF_MIO_CMIR_SEPARATION_TIME
Separation 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_PREOLVE
Value of the dual bound after presolve but before cut generation.

MSK_DINF_MIO_GMI_SEPARATION_TIME
Separation time for GMI cuts.

MSK_DINF_MIO_IMPLIED_BOUND_TIME
Separation time for implied bound cuts.

MSK_DINF_MIO_KNAPSACK_COVER_SEPARATION_TIME

Separation 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_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 node relaxation

MSK_DINF_MIO_ROOT PRESOLVE_TIME

Time spent presolving the problem at the root node.

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 occurring 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. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_BAS_DVIOLCON
Maximal dual bound violation for x^c in the basic solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_BAS_DVIOLVAR
Maximal dual bound violation for x^x in the basic solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_BAS_NRM_BARX
Infinity norm of \bar{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. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_BAS_PVIOLCON
Maximal primal bound violation for x^c in the basic solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_BAS_PVIOLVAR
Maximal primal bound violation for x^x in the basic solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_NRM_BARX
Infinity norm of \bar{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. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLBARVAR
Maximal primal bound violation for \bar{X} in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLCON
Maximal primal bound violation for x^c in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLCONES
Maximal primal violation for primal conic constraints in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLITG
Maximal violation for the integer constraints in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITG_PVIOLVAR
Maximal primal bound violation for x^x in the integer solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DUAL_OBJ
Dual objective value of the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DVIOLBARVAR
Maximal dual bound violation for \bar{X} in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DVIOLCON
Maximal dual bound violation for x^c in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DVIOLCONES
Maximal dual violation for dual conic constraints in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_DVIOLVAR
Maximal dual bound violation for x^x in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_NRM_BARS
Infinity norm of \bar{S} in the interior-point solution.

MSK_DINF_SOL_ITR_NRM_BARX
Infinity norm of \bar{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. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_PVIOLBARVAR
Maximal primal bound violation for \bar{X} in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_PVIOLCON
Maximal primal bound violation for x^c in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_PVIOLCONES
Maximal primal violation for primal conic constraints in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_SOL_ITR_PVIOLVAR

Maximal primal bound violation for x^x in the interior-point solution. Updated if *MSK_IPAR_AUTO_UPDATE_SOL_INFO* is set .

MSK_DINF_TO_CONIC_TIME

Time spent in the last to conic reformulation.

9.5.21 License feature

MSK_FEATURE_PTS

Base system.

MSK_FEATURE_PTON

Conic extension.

9.5.22 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_ANZ

Number of non-zero entries in the constraint matrix of the problem to be solved by the mixed-integer optimizer.

MSK_LIINF_MIO_INTPNT_ITER

Number of interior-point iterations performed by the mixed-integer optimizer.

MSK_LIINF_MIO_PRE SOLVED_ANZ

Number of non-zero entries in the constraint matrix of the problem after the mixed-integer optimizer's presolve.

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.

9.5.23 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.

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_CLIQUETABLE_SIZE
 Size of the clique table.

MSK_IINF_MIO_CONSTRUCT_SOLUTION
 This item informs if **MOSEK** constructed an initial integer feasible solution.

- -1: tried, but failed,
- 0: no partial solution supplied by the user,
- 1: constructed feasible solution.

MSK_IINF_MIO_NODE_DEPTH
 Depth of the last node solved.

MSK_IINF_MIO_NUM_ACTIVE_NODES
 Number of active branch and bound nodes.

MSK_IINF_MIO_NUM_BRANCH
 Number of branches performed during the optimization.

MSK_IINF_MIO_NUM_CLIQUETABLE 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 have 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_NUMBIN
Number of binary variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMBINCONEVAR
Number of binary cone variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMCON
Number of constraints in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMCONE
Number of cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMCONEVAR
Number of cone variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMCONT
Number of continuous variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMCONTCONEVAR
Number of continuous cone variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMDEXPCONES
Number of dual exponential cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMDPOWCONES
Number of dual power cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMINT
Number of integer variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMINTCONEVAR
Number of integer cone variables in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMPEXPONES
Number of primal exponential cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMPPOWCONES
Number of primal power cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMQCONES
Number of quadratic cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMRQCONES
Number of rotated quadratic cones in the problem to be solved by the mixed-integer optimizer.

MSK_IINF_MIO_NUMVAR
Number of variables in the problem to be 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 after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMBINCONEVAR
Number of binary cone variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMCON
Number of constraints in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMCONE
Number of cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMCONEVAR
Number of cone variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMCONT
Number of continuous variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMCONTCONEVAR
Number of continuous cone variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMDEXPCONES
Number of dual exponential cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMDPOWCONES
Number of dual power cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMINT
Number of integer variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRESOLVED_NUMINTCONEVAR
Number of integer cone variables in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMPEXP CONES
Number of primal exponential cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMPPOW CONES
Number of primal power cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMQ CONES
Number of quadratic cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMRQ CONES
Number of rotated quadratic cones in the problem after the mixed-integer optimizer's presolve.

MSK_IINF_MIO_PRE SOLVED_NUMVAR
Number of variables in the problem after the mixed-integer optimizer's presolve.

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_PURIFY_DUAL_SUCCESS
Is nonzero if the dual solution is purified.

MSK_IINF_PURIFY_PRIMAL_SUCCESS
Is nonzero if the primal solution is purified.

MSK_IINF_RD_NUMBARVAR
Number of symmetric 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_PROTOTYPE
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 indicate that memory fragmentation may occur.

9.5.24 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.

9.5.25 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 write.

9.5.26 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.

9.5.27 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.

9.5.28 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.

9.5.29 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_PSEUDO

The optimizer employs selects the node based on a pseudo cost estimate.

9.5.30 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.

9.5.31 Objective sense types

MSK_OBJECTIVE_SENSE_MINIMIZE

The problem should be minimized.

MSK_OBJECTIVE_SENSE_MAXIMIZE

The problem should be maximized.

9.5.32 On/off

MSK_ON

Switch the option on.

MSK_OFF

Switch the option off.

9.5.33 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.

9.5.34 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.

9.5.35 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.

9.5.36 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.

9.5.37 Problem data items

MSK_PI_VAR
Item is a variable.

MSK_PI_CON
Item is a constraint.

MSK_PI_CONE
Item is a cone.

9.5.38 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_CONIC
A conic optimization.

MSK_PROBTYPE_MIXED
General nonlinear constraints and conic constraints. This combination can not be solved by **MOSEK**.

9.5.39 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_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.

9.5.40 XML writer output mode

MSK_WRITE_XML_MODE_ROW

Write in row order.

MSK_WRITE_XML_MODE_COL

Write in column order.

9.5.41 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.

9.5.42 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.

9.5.43 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.

9.5.44 Sensitivity types

MSK_SENSITIVITY_TYPE_BASIS

Basis sensitivity analysis is performed.

9.5.45 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.

9.5.46 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.

9.5.47 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_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_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.

9.5.48 Solution types

MSK_SOL_BAS

The basic solution.

MSK_SOL_ITR

The interior solution.

MSK_SOL_ITG

The integer solution.

9.5.49 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.

9.5.50 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.

9.5.51 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 be employed when choosing the bounds on the nonlinear variables. In particular very tight bounds should be avoided.

9.5.52 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.

9.5.53 Integer values

`MSK_MAX_STR_LEN`

Maximum string length allowed in **MOSEK**.

`MSK_LICENSE_BUFFER_LENGTH`

The length of a license key buffer.

9.5.54 Variable types

`MSK_VAR_TYPE_CONT`

Is a continuous variable.

`MSK_VAR_TYPE_INT`

Is an integer variable.

Chapter 10

Supported File Formats

MOSEK supports a range of problem and solution formats listed in [Table 10.1](#) and [Table 10.2](#). 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

Table 10.1: List of supported file formats for optimization problems. The column *Conic* refers to conic problems involving the quadratic, rotated quadratic, power or exponential cone. The last two columns indicate if the format supports solutions and optimizer parameters.

Format Type	Ext.	Binary/Text	LP	QO	Conic	SDP	Sol	Param
<i>LP</i>	lp	plain text	X	X				
<i>MPS</i>	mps	plain text	X	X	X			
<i>OPF</i>	opf	plain text	X	X	X		X	X
<i>PTF</i>	ptf	plain text	X	X	X	X	X	
<i>CBF</i>	cbf	plain text	X		X	X		
<i>Task format</i>	task	binary	X	X	X	X	X	X
<i>Jtask format</i>	jtask	text	X	X	X	X	X	X

Solution formats

Table 10.2: List of supported solution formats.

Format Type	Ext.	Binary/Text	Description
<i>SOL</i>	sol	plain text	Interior Solution
	bas	plain text	Basic Solution
	int	plain text	Integer
<i>Jsol format</i>	jsol	text	Solution

Compression

MOSEK supports GZIP and Zstandard compression. Problem files with extension `.gz` (for GZIP) and `.zst` (for Zstandard) 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.

10.1 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 of the form

$$\begin{array}{ll} \text{minimize/maximize} & c^T x + \frac{1}{2} q^o(x) \\ \text{subject to} & \begin{array}{ll} l^c \leq Ax + \frac{1}{2} q(x) \leq u^c, \\ l^x \leq x \leq u^x, \\ x_{\mathcal{J}} \text{ integer,} \end{array} \end{array}$$

where

- $x \in \mathbb{R}^n$ is the vector of decision variables.
- $c \in \mathbb{R}^n$ is the linear term in the objective.
- $q^o : \mathbb{R}^n \rightarrow \mathbb{R}$ 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

$$Q^i = (Q^i)^T.$$

- $\mathcal{J} \subseteq \{1, 2, \dots, n\}$ is an index set of the integer constrained variables.

10.1.1 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.

$$c^T x + \frac{1}{2} x^T Q x.$$

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 (`[]/2`) 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

$$\text{minimize } 4x_1 + x_2 - 0.1 \cdot x_3 + \frac{1}{2}(x_1^2 + 2.1 \cdot x_1 \cdot x_2)$$

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 \leq) may be any of $<$, \leq , $=$, $>$, \geq ($<$ and \leq 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 per line, but **MOSEK** supports defining ranged constraints by using double-colon ($::$) instead of a single-colon ($:$) after the constraint name, i.e.

$$-5 \leq x_1 + x_2 \leq 5 \quad (10.1)$$

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 (10.1) may be written as

$$x_1 + x_2 - sl_1 = 0, \quad -5 \leq sl_1 \leq 5.$$

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
```

10.1.2 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
```

10.1.3 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.

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).

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 lose their uniqueness and change the problem.

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.

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 set the parameter `MSK_IPAR_WRITE_LP_STRICT_FORMAT` to `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 set the parameter `MSK_IPAR_WRITE_GENERIC_NAMES` to `MSK_ON` which will cause all names to be renamed systematically in the output file.

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` 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.
- `MSK_IPAR_WRITE_LP_TERMS_PER_LINE` 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).

10.2 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].

10.2.1 MPS File Structure

The version of the MPS format supported by **MOSEK** allows specification of an optimization problem of the form

$$\begin{aligned} \text{maximize/minimize} \quad & c^T x + q_0(x) \\ l^c \leq \quad & Ax + q(x) \leq u^c, \\ l^x \leq \quad & x \leq u^x, \\ & x \in \mathcal{K}, \\ & x_{\mathcal{J}} \text{ integer}, \end{aligned} \tag{10.2}$$

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. The same applies to q_0 .

- \mathcal{K} is a convex cone.
- $\mathcal{J} \subseteq \{1, 2, \dots, n\}$ is an index set of the integer-constrained variables.
- c is the vector of objective coefficients.

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]          [cname2]  [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

$[+|-]XXXXXXXX.XXXXXX[e|E][+|-]XXX$

where

$X = [0|1|2|3|4|5|6|7|8|9].$

- 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. 10.2.5](#) 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.

NAME (optional)

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. `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 a 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 values E, G, L, or N with the following interpretation:

Constraint type	l_i^c	u_i^c
E (equal)	finite	$= l_i^c$
G (greater)	finite	∞
L (lower)	$-\infty$	finite
N (none)	$-\infty$	∞

In the MPS format the 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 , unless something else was specified in the section OBJNAME.

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
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 + v_1 $
L	— or +	$u_i^c - v_1 $	
N			

Another constraint bound can optionally be modified using [cname2] and [value2] the same way.

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 terms belong. 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_{kj}^i 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

$$\begin{array}{ll} \text{minimize} & -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) \\ \text{subject to} & x_1 + x_2 + x_3 \geq 1, \\ & x \geq 0 \end{array}$$

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 stores all the nonzeros coefficients, without taking advantage of the symmetry of the Q matrix.
- QUADOBJ stores 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 appear 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 $1/2x^T Qx$.

The example

$$\begin{aligned}
 &\text{minimize} && -x_2 + \frac{1}{2}(2x_1^2 - 2x_1x_3 + 0.2x_2^2 + 2x_3^2) \\
 &\text{subject to} && x_1 + x_2 + x_3 \geq 1, \\
 &&& x \geq 0
 \end{aligned}$$

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

```

(continues on next page)

(continued from previous page)

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.

QMATRIX (optional)

A QMATRIX section allows to specify the quadratic part of a given constraint. Within the QMATRIX 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_{kj}^i is assigned the value given by [value1]. Moreover, the quadratic term is represented as $1/2x^T Qx$.

The example

$$\begin{aligned}
 &\text{minimize} && x_2 \\
 &\text{subject to} && x_1 + x_2 + x_3 \geq 1, \\
 &&& \frac{1}{2}(-2x_1x_3 + 0.2x_2^2 + 2x_3^2) \leq 10, \\
 &&& x \geq 0
 \end{aligned}$$

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

(continues on next page)

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 QCMATRIXs please note that:

- Only one QCMATRIX is allowed for each constraint.
- The QCMATRIXs 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.
- QCMATRIX does not exploit the symmetry of Q : an off-diagonal entry (i, j) should appear twice.

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 for which the 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

Here v_1 is the value specified by [value1].

CSECTION (optional)

The purpose of the CSECTION is to specify the conic constraint

$$x \in \mathcal{K}$$

in (10.2). It is assumed that \mathcal{K} satisfies the following requirements. Let

$$x^t \in \mathbb{R}^{n^t}, \quad t = 1, \dots, k$$

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

$$\mathcal{K} := \{x \in \mathbb{R}^n : x^t \in \mathcal{K}_t, \quad t = 1, \dots, k\}$$

where \mathcal{K}_t must have one of the following forms:

- \mathbb{R} set:

$$\mathcal{K}_t = \mathbb{R}^{n^t}.$$

- Zero cone:

$$\mathcal{K}_t = \{0\} \subseteq \mathbb{R}^{n^t}. \quad (10.3)$$

- Quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1 \geq \sqrt{\sum_{j=2}^{n^t} x_j^2} \right\}. \quad (10.4)$$

- Rotated quadratic cone:

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : 2x_1x_2 \geq \sum_{j=3}^{n^t} x_j^2, \quad x_1, x_2 \geq 0 \right\}. \quad (10.5)$$

- Primal exponential cone:

$$\mathcal{K}_t = \{x \in \mathbb{R}^3 : x_1 \geq x_2 \exp(x_3/x_2), \quad x_1, x_2 \geq 0\}. \quad (10.6)$$

- Primal power cone (with parameter $0 < \alpha < 1$):

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : x_1^\alpha x_2^{1-\alpha} \geq \sqrt{\sum_{j=3}^{n^t} x_j^2}, \quad x_1, x_2 \geq 0 \right\}. \quad (10.7)$$

- Dual exponential cone:

$$\mathcal{K}_t = \{x \in \mathbb{R}^3 : x_1 \geq -x_3 e^{-1} \exp(x_2/x_3), \quad x_3 \leq 0, x_1 \geq 0\}. \quad (10.8)$$

- Dual power cone (with parameter $0 < \alpha < 1$):

$$\mathcal{K}_t = \left\{ x \in \mathbb{R}^{n^t} : \left(\frac{x_1}{\alpha}\right)^\alpha \left(\frac{x_2}{1-\alpha}\right)^{1-\alpha} \geq \sqrt{\sum_{j=3}^{n^t} x_j^2}, \quad x_1, x_2 \geq 0 \right\}. \quad (10.9)$$

In general, membership in the \mathbb{R} set is not specified. If a variable is not a member of any other cone then it is assumed to be a member of the \mathbb{R} cone.

Next, let us study an example. Assume that the power cone

$$x_4^{1/3} x_5^{2/3} \geq |x_8|$$

and the rotated quadratic cone

$$2x_3x_7 \geq x_1^2 + x_0^2, \quad x_3, x_7 \geq 0,$$

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
*234567890123456789012345678901234567890123456789012345678901234567890						
CSECTION	konea	3e-1		PPOW		
x4						
x5						
x8						
CSECTION	koneb	0.0		RQUAD		
x7						
x3						
x1						
x0						

In general, a CSECTION header has the format

CSECTION	[kname1]	[value1]	[ktype]
----------	----------	----------	---------

where the requirements for each field are as follows:

Field	Starting Position	Max Width	Required	Description
[kname1]	15	8	Yes	Name of the cone
[value1]	25	12	No	Cone parameter
[ktype]	40		Yes	Type of the cone.

The possible cone type keys are:

[ktype]	Members	[value1]	Interpretation.
ZERO	≥ 0	unused	Zero cone (10.3).
QUAD	≥ 1	unused	Quadratic cone (10.4).
RQUAD	≥ 2	unused	Rotated quadratic cone (10.5).
PEXP	3	unused	Primal exponential cone (10.6).
PPOW	≥ 2	α	Primal power cone (10.7).
DEXP	3	unused	Dual exponential cone (10.8).
DPOW	≥ 2	α	Dual power cone (10.9).

A record in the CSECTION has the format

[vname1]

where the requirements for each field are

Field	Starting Position	Max Width	required	Description
[vname1]	5	8	Yes	A valid variable name

A variable must occur in at most one CSECTION.

ENDATA

This keyword denotes the end of the MPS file.

10.2.2 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

- 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.

10.2.3 General Limitations

- An MPS file should be an ASCII file.

10.2.4 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.

10.2.5 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, a name must not contain any blanks.

Moreover, by default a line in the MPS file must not contain more than 1024 characters. By modifying the parameter `MSK_IPAR_READ_MPS_WIDTH` an arbitrary large line width will be accepted.

The free MPS format is default. To change to the strict and other formats use the parameter `MSK_IPAR_READ_MPS_FORMAT`.

10.3 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.

10.3.1 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 [/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]
```

10.3.2 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 subsections `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 subsections `b` (linear bounds on variables) and `cone` (cones).

[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]

Specifies a cone. A cone is defined as a sequence of variables which belong to a single unique cone. The supported cone types are:

- **quad**: a quadratic cone of n variables x_1, \dots, x_n defines a constraint of the form

$$x_1^2 \geq \sum_{i=2}^n x_i^2, \quad x_1 \geq 0.$$

- **rquad**: a rotated quadratic cone of n variables x_1, \dots, x_n defines a constraint of the form

$$2x_1x_2 \geq \sum_{i=3}^n x_i^2, \quad x_1, x_2 \geq 0.$$

- **pexp**: primal exponential cone of 3 variables x_1, x_2, x_3 defines a constraint of the form

$$x_1 \geq x_2 \exp(x_3/x_2), \quad x_1, x_2 \geq 0.$$

- **ppow** with parameter $0 < \alpha < 1$: primal power cone of n variables x_1, \dots, x_n defines a constraint of the form

$$x_1^\alpha x_2^{1-\alpha} \geq \sqrt{\sum_{j=3}^n x_j^2}, \quad x_1, x_2 \geq 0.$$

- **dexp**: dual exponential cone of 3 variables x_1, x_2, x_3 defines a constraint of the form

$$x_1 \geq -x_3 e^{-1} \exp(x_2/x_3), \quad x_3 \leq 0, x_1 \geq 0.$$

- **dpow** with parameter $0 < \alpha < 1$: dual power cone of n variables x_1, \dots, x_n defines a constraint of the form

$$\left(\frac{x_1}{\alpha}\right)^\alpha \left(\frac{x_2}{1-\alpha}\right)^{1-\alpha} \geq \sqrt{\sum_{j=3}^n x_j^2}, \quad x_1, x_2 \geq 0.$$

- **zero**: zero cone of n variables x_1, \dots, x_n defines a constraint of the form

$$x_1 = \dots = x_n = 0$$

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 rquad] x,y,z,w [/cone] # rotated quadratic cone
[cone ppow '3e-01' 'a'] x1, x2, x3 [/cone] # power cone with alpha=1/3 and name 'a'
[/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.

[integer]

This contains a space-separated list of variables and defines the constraint that the listed variables must be integer-valued.

[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 is defined as follows:

```
[hint ITEM] value [/hint]
```

The hints recognized by **MOSEK** are:

- `numvar` (number of variables),
- `numcon` (number of linear/quadratic constraints),
- `numanz` (number of linear non-zeros in constraints),
- `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]
```

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).
- **lv1** Defines the level of the item.
- **s1** 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**, **lv1**, **s1** and **su**. Items **s1** and **su** are not required for integer solutions.

A [con] section should always contain **sk**, **lv1**, **s1**, **su** and **y**.

An example of a solution section

```
[solution basic status=UNKNOWN]
[var x0] sk=LOW    lv1=5.0    [/var]
[var x1] sk=UPR    lv1=10.0   [/var]
[var x2] sk=SUPBAS lv1=2.0    s1=1.5 su=0.0 [/var]

[con c0] sk=LOW    lv1=3.0 y=0.0 [/con]
[con c0] sk=UPR    lv1=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.

10.3.3 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]+)?
```

10.3.4 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"
```

10.3.5 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]
```

10.3.6 Writing OPF Files from MOSEK

To write an OPF file then make sure the file extension is .opf.

Then modify the following parameters to define what the file should contain:

<i>MSK_IPAR_OPF_WRITE_SOL_BAS</i>	Include basic solution, if defined.
<i>MSK_IPAR_OPF_WRITE_SOL_ITG</i>	Include integer solution, if defined.
<i>MSK_IPAR_OPF_WRITE_SOL_ITR</i>	Include interior solution, if defined.
<i>MSK_IPAR_OPF_WRITE_SOLUTIONS</i>	Include solutions if they are defined. If this is off, no solutions are included.
<i>MSK_IPAR_OPF_WRITE_HEADER</i>	Include a small header with comments.
<i>MSK_IPAR_OPF_WRITE_PROBLEM</i>	Include the problem itself — objective, constraints and bounds.
<i>MSK_IPAR_OPF_WRITE_PARAMETERS</i>	Include all parameter settings.
<i>MSK_IPAR_OPF_WRITE_HINTS</i>	Include hints about the size of the problem.

10.3.7 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:

$$\begin{aligned}
&\text{maximize} && 3x_0 + 1x_1 + 5x_2 + 1x_3 \\
&\text{subject to} && 3x_0 + 1x_1 + 2x_2 = 30, \\
& && 2x_0 + 1x_1 + 3x_2 + 1x_3 \geq 15, \\
& && 2x_1 + 3x_3 \leq 25,
\end{aligned}$$

having the bounds

$$\begin{aligned}
0 &\leq x_0 \leq \infty, \\
0 &\leq x_1 \leq 10, \\
0 &\leq x_2 \leq \infty, \\
0 &\leq x_3 \leq \infty.
\end{aligned}$$

In the OPF format the example is displayed as shown in [Listing 10.1](#).

Listing 10.1: 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]
```

(continues on next page)

```

[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

$$\begin{aligned}
 &\text{minimize} && x_1^2 + 0.1x_2^2 + x_3^2 - x_1x_3 - x_2 \\
 &\text{subject to} && 1 \leq x_1 + x_2 + x_3, \\
 &&& x \geq 0.
 \end{aligned}$$

This can be formulated in `opf` as shown below.

Listing 10.2: 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:

$$\begin{aligned}
&\text{minimize} && x_3 + x_4 + x_5 \\
&\text{subject to} && x_0 + x_1 + 2x_2 = 1, \\
& && x_0, x_1, x_2 \geq 0, \\
& && x_3 \geq \sqrt{x_0^2 + x_1^2}, \\
& && 2x_4x_5 \geq x_2^2.
\end{aligned}$$

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 10.3](#).

Listing 10.3: 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:

$$\begin{array}{llll} \text{maximize} & x_0 + 0.64x_1 & & \\ \text{subject to} & 50x_0 + 31x_1 & \leq & 250, \\ & 3x_0 - 2x_1 & \geq & -4, \\ & x_0, x_1 \geq 0 & & \text{and integer} \end{array}$$

This can be implemented in OPF with the file in [Listing 10.4](#).

Listing 10.4: 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]
```

10.4 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.

10.4.1 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:

$$\begin{aligned} & \min / \max && g^{obj} \\ \text{s.t.} &&& \begin{aligned} g_i &\in \mathcal{K}_i, & i &\in \mathcal{I}, \\ G_i &\in \mathcal{K}_i, & i &\in \mathcal{I}^{PSD}, \\ x_j &\in \mathcal{K}_j, & j &\in \mathcal{J}, \\ \bar{X}_j &\in \mathcal{K}_j, & j &\in \mathcal{J}^{PSD}. \end{aligned} \end{aligned} \tag{10.10}$$

- **Variables** are either scalar variables, x_j for $j \in \mathcal{J}$, or variables, \bar{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 $i \in \mathcal{I}^{PSD}$

$$\begin{aligned} g_i &= \sum_{j \in \mathcal{J}^{PSD}} \langle F_{ij}, X_j \rangle + \sum_{j \in \mathcal{J}} a_{ij} x_j + b_i, \\ G_i &= \sum_{j \in \mathcal{J}} x_j H_{ij} + D_i. \end{aligned}$$

- 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

$$\{x \in \mathbb{R}^n \mid x_j \geq 0 \text{ for } j = 1, \dots, n\}, \text{ for } n \geq 1.$$

- **Negative orthant** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j \leq 0 \text{ for } j = 1, \dots, n\}, \text{ for } n \geq 1.$$

- **Fixpoint zero** - A cone in the linear family defined by

$$\{x \in \mathbb{R}^n \mid x_j = 0 \text{ for } j = 1, \dots, n\}, \text{ for } n \geq 1.$$

- **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

$$\left\{ \begin{pmatrix} p \\ q \\ x \end{pmatrix} \in \mathbb{R} \times \mathbb{R} \times \mathbb{R}^{n-2}, 2pq \geq x^T x, p \geq 0, q \geq 0 \right\}, \text{ for } n \geq 3.$$

10.4.2 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 separator 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.

10.4.3 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, $\{0, 1, \dots\}$, 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

$$x \in \mathcal{K}_1^{n_1} \times \mathcal{K}_2^{n_2} \times \dots \times \mathcal{K}_k^{n_k}$$

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 10.3. 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

$$g \in \mathcal{K}_1^{m_1} \times \mathcal{K}_2^{m_2} \times \dots \times \mathcal{K}_k^{m_k}$$

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 10.3](#).

- **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 10.3: 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

10.4.4 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 10.3](#)), 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 10.3), 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

10.4.5 CBF Format Examples

Minimal Working Example

The conic optimization problem (10.11), has three variables in a quadratic cone - first one is integer - and an affine expression in domain 0 (equality constraint).

$$\begin{aligned} & \text{minimize} && 5.1 x_0 \\ & \text{subject to} && 6.2 x_1 + 7.3 x_2 - 8.4 \in \{0\} \\ & && x \in \mathcal{Q}^3, x_0 \in \mathbb{Z}. \end{aligned} \tag{10.11}$$

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.

```
OBJCOORD
```

```
1
```

```
0 5.1
```

```
ACCOORD
```

```
2
```

```
0 1 6.2
```

```
0 2 7.3
```

```
BCCOORD
```

```
1
```

```
0 -8.4
```

This concludes the example.

Mixing Linear, Second-order and Semidefinite Cones

The conic optimization problem (10.12), has a semidefinite cone, a quadratic cone over unordered subindices, and two equality constraints.

$$\begin{aligned} \text{minimize} \quad & \left\langle \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}, X_1 \right\rangle + x_1 \\ \text{subject to} \quad & \left\langle \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 = 1.0, \\ & \left\langle \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, X_1 \right\rangle + x_0 + x_2 = 0.5, \\ & x_1 \geq \sqrt{x_0^2 + x_2^2}, \\ & X_1 \succeq \mathbf{0}. \end{aligned} \tag{10.12}$$

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
```

(continues on next page)

```

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

```

(continues on next page)

(continued from previous page)

```
# 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.

$$\begin{aligned} \text{minimize} \quad & \left\langle \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, X_1 \right\rangle + x_1 + x_2 + 1 \\ \text{subject to} \quad & \left\langle \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, X_1 \right\rangle - x_1 - x_2 \geq 0.0, \\ & x_1 \begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix} + x_2 \begin{bmatrix} 3 & 1 \\ 1 & 0 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \preceq \mathbf{0}, \\ & X_1 \succeq \mathbf{0}. \end{aligned} \tag{10.13}$$

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
```

(continues on next page)

```

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:
#   |  $a[0,0] = -1.0$ 
#   |  $a[0,1] = -1.0$ 
ACCOORD
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 (10.14), is defined for a sequence of objectives such that hotstarting from one to the next might be advantages.

$$\begin{aligned} & \text{maximize}_k && g_k^{obj} \\ & \text{subject to} && 50x_0 + 31 \leq 250, \\ & && 3x_0 - 2x_1 \geq -4, \\ & && x \in \mathbb{R}_+^2, \end{aligned} \tag{10.14}$$

given,

1. $g_0^{obj} = x_0 + 0.64x_1$.
2. $g_1^{obj} = 1.11x_0 + 0.76x_1$.
3. $g_2^{obj} = 1.11x_0 + 0.85x_1$.

Its formulation in the CBF format is reported in Listing 10.5.

Listing 10.5: Problem (10.14) 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
```

(continues on next page)


```

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
OBJACORD
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
OBJACORD
1
1 0.85

```

10.5 The PTF Format

The PTF format is a new human-readable, natural text format. Its features and structure are similar to the *OPF* format, with the difference that the PTF format **does** support semidefinite terms.

10.5.1 The overall format

The format is indentation based, where each section is started by a head line and followed by a section body with deeper indentation than the head line. For example:

```

Header line
  Body line 1
  Body line 1
  Body line 1

```

Section can also be nested:

```

Header line A
  Body line in A
  Header line A.1
    Body line in A.1

```

(continues on next page)

```

    Body line in A.1
Body line in A

```

The indentation of blank lines is ignored, so a subsection can contain a blank line with no indentation. The character # defines a line comment and anything between the # character and the end of the line is ignored.

In a PTF file, the first section must be a **Task** section. The order of the remaining section is arbitrary, and sections may occur multiple times or not at all.

MOSEK will ignore any top-level section it does not recognize.

Names

In the description of the format we use following definitions for name strings:

```

NAME: PLAIN_NAME | QUOTED_NAME
PLAIN_NAME: [a-zA-Z_] [a-zA-Z0-9_-.!|]
QUOTED_NAME: '"' ( [^'\\r\n] | "\\" ( [\\rn] | "x" [0-9a-fA-F] [0-9a-fA-F] ) ) * '"'

```

Expressions

An expression is a sum of terms. A term is either a linear term (a coefficient and a variable name, where the coefficient can be left out if it is 1.0), or a matrix inner product.

An expression:

```

EXPR: EMPTY | [+]? TERM ( [+]? TERM ) *
TERM: LINEAR_TERM | MATRIX_TERM

```

A linear term

```

LINEAR_TERM: FLOAT? NAME

```

A matrix term

```

MATRIX_TERM: "<" FLOAT? NAME ( [+]? FLOAT? NAME ) * ";" NAME ">"

```

Here the right-hand name is the name of a (semidefinite) matrix variable, and the left-hand side is a sum of symmetric matrixes. The actual matrixes are defined in a separate section.

Expressions can span multiple lines by giving subsequent lines a deeper indentation.

For example following two section are equivalent:

```

# Everything on one line:
x1 + x2 + x3 + x4

# Split into multiple lines:
x1
+ x2
+ x3
+ x4

```

10.5.2 Task section

The first section of the file must be a **Task**. The text in this section is not used and may contain comments, or meta-information from the writer or about the content.

Format:

```
Task NAME
  Anything goes here...
```

NAME is a the task name.

10.5.3 Objective section

The **Objective** section defines the objective name, sense and function. The format:

```
"Objective" NAME?
  ( "Minimize" | "Maximize" ) EXPR
```

For example:

```
Objective 'obj'
  Minimize x1 + 0.2 x2 + < M1 ; X1 >
```

10.5.4 Constraints section

The constraints section defines a series of constraints. A constraint defines a term $A \cdot x + b \in K$. For linear constraints **A** is just one row, while for conic constraints it can be multiple rows. If a constraint spans multiple rows these can either be written inline separated by semi-colons, or each expression in a separate sub-section.

Simple linear constraints:

```
"Constraints"
NAME? "[" [-+] (FLOAT | "Inf") (";" [-+] (FLOAT | "Inf"))? "]" EXPR
```

If the brackets contain two values, they are used as upper and lower bounds. If they contain one value the constraint is an equality.

For example:

```
Constraints
'c1' [0;10] x1 + x2 + x3
[0] x1 + x2 + x3
```

Constraint blocks put the expression either in a subsection or inline. The cone type (domain) is written in the brackets, and **MOSEK** currently supports following types:

- SOC(N) Second order cone of dimension N
- RSOC(N) Rotated second order cone of dimension N
- PSD(N) Symmetric positive semidefinite cone of dimension N. This contains $N*(N+1)/2$ elements.
- PEXP Primal exponential cone of dimension 3
- DEXP Dual exponential cone of dimension 3
- PPOW(N,P) Primal power cone of dimension N with parameter P
- DPOW(N,P) Dual power cone of dimension N with parameter P
- ZERO(N) The zero-cone of dimension N.

```
"Constraints"
NAME? "[" DOMAIN "]" EXPR_LIST
```

For example:

```
Constraints
'K1' [SOC(3)] x1 + x2 ; x2 + x3 ; x3 + x1
'K2' [RSOC(3)]
    x1 + x2
    x2 + x3
    x3 + x1
```

10.5.5 Variables section

Any variable used in an expression must be defined in a variable section. The variable section defines each variable domain.

```
"Variables"
NAME "[" [-+] (FLOAT | "Inf") (";" [-+] (FLOAT | "Inf"))? "]"
NAME "[" DOMAIN "]" NAMES

For example, a linear variable
```

```
Variables
x1 [0;Inf]
```

As with constraints, members of a conic domain can be listed either inline or in a subsection:

```
Variables
k1 [SOC(3)] x1 ; x2 ; x3
k2 [RSOC(3)]
    x1
    x2
    x3
```

10.5.6 Integer section

This section contains a list of variables that are integral. For example:

```
Integer
x1 x2 x3
```

10.5.7 SymmetricMatrixes section

This section defines the symmetric matrixes used for matrix coefficients in matrix inner product terms. The section lists named matrixes, each with a size and a number of non-zeros. Only non-zeros in the lower triangular part should be defined.

```
"SymmetricMatrixes"
NAME "SYMMAT" "(" INT ")" ( "(" INT "," INT "," FLOAT ")" ) *
...
```

For example:

```
SymmetricMatrixes
M1 SYMMAT(3) (0,0,1.0) (1,1,2.0) (2,1,0.5)
M2 SYMMAT(3)
    (0,0,1.0)
    (1,1,2.0)
    (2,1,0.5)
```

10.5.8 Solutions section

Each subsection defines a solution. A solution defines for each constraint and for each variable exactly one primal value and either one (for conic domains) or two (for linear domains) dual values. The values follow the same logic as in the **MOSEK C API**. A primal and a dual solution status defines the meaning of the values primal and dual (solution, certificate, unknown, etc.)

The format is this:

```
"Solutions"
  "Solution" WHICHSQL
    "ProblemStatus" PROSTA PROSTA?
  "SolutionStatus" SOLSTA SOLSTA?
  "Objective" FLOAT FLOAT
  "Variables"
    # Linear variable status: level, slx, sux
    NAME "[" STATUS "]" FLOAT (FLOAT FLOAT)?
    # Conic variable status: level, snx
    NAME
      "[" STATUS "]" FLOAT FLOAT?
    ...
  "Constraints"
    # Linear variable status: level, slx, sux
    NAME "[" STATUS "]" FLOAT (FLOAT FLOAT)?
    # Conic variable status: level, snx
    NAME
      "[" STATUS "]" FLOAT FLOAT?
    ...
```

Following values for **WHICHSQL** are supported:

- **interior** Interior solution, the result of an interior-point solver.
- **basic** Basic solution, as produced by a simplex solver.
- **integer** Integer solution, the solution to a mixed-integer problem. This does not define a dual solution.

Following values for **PROSTA** are supported:

- **unknown** The problem status is unknown
- **feasible** The problem has been proven feasible
- **infeasible** The problem has been proven infeasible
- **illposed** The problem has been proved to be ill posed
- **infeasible_or_unbounded** The problem is infeasible or unbounded

Following values for **SOLSTA** are supported:

- **unknown** The solution status is unknown
- **feasible** The solution is feasible
- **optimal** The solution is optimal
- **infeas_cert** The solution is a certificate of infeasibility
- **illposed_cert** The solution is a certificate of illposedness

Following values for **STATUS** are supported:

- **unknown** The value is unknown
- **super_basic** The value is super basic

- `at_lower` The value is basic and at its lower bound
- `at_upper` The value is basic and at its upper bound
- `fixed` The value is basic fixed
- `infinite` The value is at infinity

10.6 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, 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:

- 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.

10.7 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`

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.

10.7.1 *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.

10.7.2 *jsol* format

It stores a problem solution. The *jsol* format contains all solutions and information items.

10.7.3 A *jtask* example

In Listing 10.6 we present a file in the *jtask* format that corresponds to the sample problem from 1o1.1p. The listing has been formatted for readability.

Listing 10.6: A formatted *jtask* file for the 1o1.1p example.

```
{
  "$schema": "http://mosek.com/json/schema#",
  "Task/INFO": {
    "taskname": "1o1",
    "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"
      ],
      "b1": [
        0.0,
        0.0,
        0.0,
        0.0
      ],
      "bu": [
        1e+30,
        1e+1,
        1e+30,

```

(continues on next page)

```

        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"
    }
}
}

```

10.8 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>

(continues on next page)

(continued from previous page)

DUAL OBJECTIVE : <dual objective value corresponding to the solution>						
CONSTRAINTS						
INDEX	NAME	AT	ACTIVITY	LOWER LIMIT	UPPER LIMIT	DUAL LOWER
?	<name>	??	<a value>	<a value>	<a value>	<a value>
VARIABLES						
INDEX	NAME	AT	ACTIVITY	LOWER LIMIT	UPPER LIMIT	DUAL LOWER
?	<name>	??	<a value>	<a value>	<a value>	<a value>
CONIC DUAL						
?	<name>	??	<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

$$l_i^c \leq \sum_{j=1}^n a_{ij}x_j \leq u_i^c, \quad (10.15)$$

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 10.4 the possible values of the status keys and their interpretation are shown.

Table 10.4: 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 (10.15).)
- **UPPER LIMIT**: the quantity u_i^c (see (10.15).)
- **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: 1o1.sol

In Listing 10.7 we show the solution file for the 1o1.opf problem.

Listing 10.7: 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      -2.49999999741653e+00
1          c2              SB 5.33333333049187e+01      1.50000000e+01      NONE
↪      2.09159033069640e-10      -0.00000000000000e+00
2          c3              UL 2.49999999842049e+01      NONE                2.
↪50000000e+01      -0.00000000000000e+00      -3.33333332895108e-01

VARIABLES
INDEX      NAME          AT ACTIVITY          LOWER LIMIT      UPPER LIMIT
↪      DUAL LOWER          DUAL UPPER
0          x1              LL 1.67020427038537e-09      0.00000000e+00      NONE
↪      -4.49999999528054e+00      -0.00000000000000e+00
1          x2              LL 2.93510446211883e-09      0.00000000e+00      1.
↪00000000e+01      -2.16666666494915e+00      6.20868657679896e-10
2          x3              SB 1.49999999899424e+01      0.00000000e+00      NONE
↪      -8.79123177245553e-10      -0.00000000000000e+00
3          x4              SB 8.33333332273115e+00      0.00000000e+00      NONE
↪      -1.69795978848200e-09      -0.00000000000000e+00

```


Bibliography

- [Naz87] J. L. Nazareth. *Computer Solution of Linear Programs*. Oxford University Press, New York, 1987.

Symbol Index

Functions

Parameters

Double parameters, 43

MSK_DPAR_ANA_SOL_INFEAS_TOL, 43
MSK_DPAR_BASIS_REL_TOL_S, 43
MSK_DPAR_BASIS_TOL_S, 43
MSK_DPAR_BASIS_TOL_X, 43
MSK_DPAR_CHECK_CONVEXITY_REL_TOL, 43
MSK_DPAR_DATA_SYM_MAT_TOL, 44
MSK_DPAR_DATA_SYM_MAT_TOL_HUGE, 44
MSK_DPAR_DATA_SYM_MAT_TOL_LARGE, 44
MSK_DPAR_DATA_TOL_AIJ_HUGE, 44
MSK_DPAR_DATA_TOL_AIJ_LARGE, 44
MSK_DPAR_DATA_TOL_BOUND_INF, 44
MSK_DPAR_DATA_TOL_BOUND_WRN, 44
MSK_DPAR_DATA_TOL_C_HUGE, 45
MSK_DPAR_DATA_TOL_CJ_LARGE, 45
MSK_DPAR_DATA_TOL_QIJ, 45
MSK_DPAR_DATA_TOL_X, 45
MSK_DPAR_INTPNT_CO_TOL_DFEAS, 45
MSK_DPAR_INTPNT_CO_TOL_INFEAS, 45
MSK_DPAR_INTPNT_CO_TOL_MU_RED, 45
MSK_DPAR_INTPNT_CO_TOL_NEAR_REL, 46
MSK_DPAR_INTPNT_CO_TOL_PFEAS, 46
MSK_DPAR_INTPNT_CO_TOL_REL_GAP, 46
MSK_DPAR_INTPNT_QO_TOL_DFEAS, 46
MSK_DPAR_INTPNT_QO_TOL_INFEAS, 46
MSK_DPAR_INTPNT_QO_TOL_MU_RED, 46
MSK_DPAR_INTPNT_QO_TOL_NEAR_REL, 47
MSK_DPAR_INTPNT_QO_TOL_PFEAS, 47
MSK_DPAR_INTPNT_QO_TOL_REL_GAP, 47
MSK_DPAR_INTPNT_TOL_DFEAS, 47
MSK_DPAR_INTPNT_TOL_DSAFE, 47
MSK_DPAR_INTPNT_TOL_INFEAS, 47
MSK_DPAR_INTPNT_TOL_MU_RED, 47
MSK_DPAR_INTPNT_TOL_PATH, 48
MSK_DPAR_INTPNT_TOL_PFEAS, 48
MSK_DPAR_INTPNT_TOL_PSAFE, 48
MSK_DPAR_INTPNT_TOL_REL_GAP, 48
MSK_DPAR_INTPNT_TOL_REL_STEP, 48
MSK_DPAR_INTPNT_TOL_STEP_SIZE, 48
MSK_DPAR_LOWER_OBJ_CUT, 49
MSK_DPAR_LOWER_OBJ_CUT_FINITE_TRH, 49
MSK_DPAR_MIO_MAX_TIME, 49
MSK_DPAR_MIO_REL_GAP_CONST, 49
MSK_DPAR_MIO_TOL_ABS_GAP, 49
MSK_DPAR_MIO_TOL_ABS_RELAX_INT, 49
MSK_DPAR_MIO_TOL_FEAS, 49

MSK_DPAR_MIO_TOL_REL_DUAL_BOUND_IMPROVEMENT,
50

MSK_DPAR_MIO_TOL_REL_GAP, 50
MSK_DPAR_OPTIMIZER_MAX_TIME, 50
MSK_DPAR_PREOLVE_TOL_ABS_LINDEP, 50
MSK_DPAR_PREOLVE_TOL_AIJ, 50
MSK_DPAR_PREOLVE_TOL_REL_LINDEP, 50
MSK_DPAR_PREOLVE_TOL_S, 50
MSK_DPAR_PREOLVE_TOL_X, 51
MSK_DPAR_QCQO_REFORMULATE_REL_DROP_TOL, 51
MSK_DPAR_SEMIDEFINITE_TOL_APPROX, 51
MSK_DPAR_SIM_LU_TOL_REL_PIV, 51
MSK_DPAR_SIMPLEX_ABS_TOL_PIV, 51
MSK_DPAR_UPPER_OBJ_CUT, 51
MSK_DPAR_UPPER_OBJ_CUT_FINITE_TRH, 51

Integer parameters, 52

MSK_IPAR_ANA_SOL_BASIS, 52
MSK_IPAR_ANA_SOL_PRINT_VIOLATED, 52
MSK_IPAR_AUTO_SORT_A_BEFORE_OPT, 52
MSK_IPAR_AUTO_UPDATE_SOL_INFO, 52
MSK_IPAR_BASIS_SOLVE_USE_PLUS_ONE, 52
MSK_IPAR_BI_CLEAN_OPTIMIZER, 52
MSK_IPAR_BI_IGNORE_MAX_ITER, 53
MSK_IPAR_BI_IGNORE_NUM_ERROR, 53
MSK_IPAR_BI_MAX_ITERATIONS, 53
MSK_IPAR_CACHE_LICENSE, 53
MSK_IPAR_CHECK_CONVEXITY, 53
MSK_IPAR_COMPRESS_STATFILE, 53
MSK_IPAR_INFEAS_GENERIC_NAMES, 54
MSK_IPAR_INFEAS_PREFER_PRIMAL, 54
MSK_IPAR_INFEAS_REPORT_AUTO, 54
MSK_IPAR_INFEAS_REPORT_LEVEL, 54
MSK_IPAR_INTPNT_BASIS, 54
MSK_IPAR_INTPNT_DIFF_STEP, 54
MSK_IPAR_INTPNT_HOTSTART, 54
MSK_IPAR_INTPNT_MAX_ITERATIONS, 55
MSK_IPAR_INTPNT_MAX_NUM_COR, 55
MSK_IPAR_INTPNT_MAX_NUM_REFINEMENT_STEPS,

55

MSK_IPAR_INTPNT_MULTI_THREAD, 55
MSK_IPAR_INTPNT_OFF_COL_TRH, 55
MSK_IPAR_INTPNT_ORDER_GP_NUM_SEEDS, 55
MSK_IPAR_INTPNT_ORDER_METHOD, 56
MSK_IPAR_INTPNT_PURIFY, 56
MSK_IPAR_INTPNT_REGULARIZATION_USE, 56
MSK_IPAR_INTPNT_SCALING, 56
MSK_IPAR_INTPNT_SOLVE_FORM, 56
MSK_IPAR_INTPNT_STARTING_POINT, 56
MSK_IPAR_LICENSE_DEBUG, 56

MSK_IPAR_LICENSE_PAUSE_TIME, 57
 MSK_IPAR_LICENSE_SUPPRESS_EXPIRE_WRNS, 57
 MSK_IPAR_LICENSE_TRH_EXPIRY_WRN, 57
 MSK_IPAR_LICENSE_WAIT, 57
 MSK_IPAR_LOG, 57
 MSK_IPAR_LOG_ANA_PRO, 57
 MSK_IPAR_LOG_BI, 57
 MSK_IPAR_LOG_BI_FREQ, 58
 MSK_IPAR_LOG_CHECK_CONVEXITY, 58
 MSK_IPAR_LOG_CUT_SECOND_OPT, 58
 MSK_IPAR_LOG_EXPAND, 58
 MSK_IPAR_LOG_FEAS_REPAIR, 58
 MSK_IPAR_LOG_FILE, 58
 MSK_IPAR_LOG_INCLUDE_SUMMARY, 59
 MSK_IPAR_LOG_INFEAS_ANA, 59
 MSK_IPAR_LOG_INTPT, 59
 MSK_IPAR_LOG_LOCAL_INFO, 59
 MSK_IPAR_LOG_MIO, 59
 MSK_IPAR_LOG_MIO_FREQ, 59
 MSK_IPAR_LOG_ORDER, 60
 MSK_IPAR_LOG_PREOLVE, 60
 MSK_IPAR_LOG_RESPONSE, 60
 MSK_IPAR_LOG_SENSITIVITY, 60
 MSK_IPAR_LOG_SENSITIVITY_OPT, 60
 MSK_IPAR_LOG_SIM, 60
 MSK_IPAR_LOG_SIM_FREQ, 60
 MSK_IPAR_LOG_SIM_MINOR, 61
 MSK_IPAR_LOG_STORAGE, 61
 MSK_IPAR_MAX_NUM_WARNINGS, 61
 MSK_IPAR_MIO_BRANCH_DIR, 61
 MSK_IPAR_MIO_CONIC_OUTER_APPROXIMATION, 61
 MSK_IPAR_MIO_CUT_CLIQUE, 61
 MSK_IPAR_MIO_CUT_CMIR, 61
 MSK_IPAR_MIO_CUT_GMI, 62
 MSK_IPAR_MIO_CUT IMPLIED_BOUND, 62
 MSK_IPAR_MIO_CUT_KNAPSACK_COVER, 62
 MSK_IPAR_MIO_CUT_SELECTION_LEVEL, 62
 MSK_IPAR_MIO_FEASPUMP_LEVEL, 62
 MSK_IPAR_MIO_HEURISTIC_LEVEL, 62
 MSK_IPAR_MIO_MAX_NUM_BRANCHES, 63
 MSK_IPAR_MIO_MAX_NUM_RELAXS, 63
 MSK_IPAR_MIO_MAX_NUM_ROOT_CUT_ROUNDS, 63
 MSK_IPAR_MIO_MAX_NUM_SOLUTIONS, 63
 MSK_IPAR_MIO_MODE, 63
 MSK_IPAR_MIO_NODE_OPTIMIZER, 63
 MSK_IPAR_MIO_NODE_SELECTION, 64
 MSK_IPAR_MIO_PERSPECTIVE_REFORMULATE, 64
 MSK_IPAR_MIO_PROBING_LEVEL, 64
 MSK_IPAR_MIO_PROPAGATE_OBJECTIVE_CONSTRAINT, 64
 MSK_IPAR_MIO_RINS_MAX_NODES, 64
 MSK_IPAR_MIO_ROOT_OPTIMIZER, 64
 MSK_IPAR_MIO_ROOT_REPEAT_PREOLVE_LEVEL, 65
 MSK_IPAR_MIO_SEED, 65
 MSK_IPAR_MIO_VB_DETECTION_LEVEL, 65
 MSK_IPAR_MT_SPINCOUNT, 65
 MSK_IPAR_NUM_THREADS, 65
 MSK_IPAR_OPF_WRITE_HEADER, 66
 MSK_IPAR_OPF_WRITE_HINTS, 66
 MSK_IPAR_OPF_WRITE_LINE_LENGTH, 66
 MSK_IPAR_OPF_WRITE_PARAMETERS, 66
 MSK_IPAR_OPF_WRITE_PROBLEM, 66
 MSK_IPAR_OPF_WRITE_SOL_BAS, 66
 MSK_IPAR_OPF_WRITE_SOL_ITG, 66
 MSK_IPAR_OPF_WRITE_SOL_ITR, 67
 MSK_IPAR_OPF_WRITE_SOLUTIONS, 67
 MSK_IPAR_OPTIMIZER, 67
 MSK_IPAR_PARAM_READ_CASE_NAME, 67
 MSK_IPAR_PARAM_READ_IGN_ERROR, 67
 MSK_IPAR_PREOLVE_ELIMINATOR_MAX_FILL, 67
 MSK_IPAR_PREOLVE_ELIMINATOR_MAX_NUM_TRIES, 67
 MSK_IPAR_PREOLVE_LEVEL, 68
 MSK_IPAR_PREOLVE_LINDEP_ABS_WORK_TRH, 68
 MSK_IPAR_PREOLVE_LINDEP_REL_WORK_TRH, 68
 MSK_IPAR_PREOLVE_LINDEP_USE, 68
 MSK_IPAR_PREOLVE_MAX_NUM_PASS, 68
 MSK_IPAR_PREOLVE_MAX_NUM_REDUCTIONS, 68
 MSK_IPAR_PREOLVE_USE, 68
 MSK_IPAR_PRIMAL_REPAIR_OPTIMIZER, 69
 MSK_IPAR_PTF_WRITE_TRANSFORM, 69
 MSK_IPAR_READ_DEBUG, 69
 MSK_IPAR_READ_KEEP_FREE_CON, 69
 MSK_IPAR_READ_LP_DROP_NEW_VARS_IN_BOU, 69
 MSK_IPAR_READ_LP_QUOTED_NAMES, 69
 MSK_IPAR_READ_MPS_FORMAT, 69
 MSK_IPAR_READ_MPS_WIDTH, 70
 MSK_IPAR_READ_TASK_IGNORE_PARAM, 70
 MSK_IPAR_REMOVE_UNUSED_SOLUTIONS, 70
 MSK_IPAR_SENSITIVITY_ALL, 70
 MSK_IPAR_SENSITIVITY_OPTIMIZER, 70
 MSK_IPAR_SENSITIVITY_TYPE, 70
 MSK_IPAR_SIM_BASIS_FACTOR_USE, 70
 MSK_IPAR_SIM_DEGEN, 71
 MSK_IPAR_SIM_DUAL_CRASH, 71
 MSK_IPAR_SIM_DUAL_PHASEONE_METHOD, 71
 MSK_IPAR_SIM_DUAL_RESTRICT_SELECTION, 71
 MSK_IPAR_SIM_DUAL_SELECTION, 71
 MSK_IPAR_SIM_EXPLOIT_DUPVEC, 71
 MSK_IPAR_SIM_HOTSTART, 72
 MSK_IPAR_SIM_HOTSTART_LU, 72
 MSK_IPAR_SIM_MAX_ITERATIONS, 72
 MSK_IPAR_SIM_MAX_NUM_SETBACKS, 72
 MSK_IPAR_SIM_NON_SINGULAR, 72
 MSK_IPAR_SIM_PRIMAL_CRASH, 72
 MSK_IPAR_SIM_PRIMAL_PHASEONE_METHOD, 73
 MSK_IPAR_SIM_PRIMAL_RESTRICT_SELECTION, 73
 MSK_IPAR_SIM_PRIMAL_SELECTION, 73
 MSK_IPAR_SIM_REFACTOR_FREQ, 73
 MSK_IPAR_SIM_REFORMULATION, 73
 MSK_IPAR_SIM_SAVE_LU, 73
 MSK_IPAR_SIM_SCALING, 74
 MSK_IPAR_SIM_SCALING_METHOD, 74
 MSK_IPAR_SIM_SEED, 74
 MSK_IPAR_SIM SOLVE_FORM, 74
 MSK_IPAR_SIM_STABILITY_PRIORITY, 74

MSK_IPAR_SIM_SWITCH_OPTIMIZER, 74
 MSK_IPAR_SOL_FILTER_KEEP_BASIC, 74
 MSK_IPAR_SOL_FILTER_KEEP_RANGED, 75
 MSK_IPAR_SOL_READ_NAME_WIDTH, 75
 MSK_IPAR_SOL_READ_WIDTH, 75
 MSK_IPAR_SOLUTION_CALLBACK, 75
 MSK_IPAR_TIMING_LEVEL, 75
 MSK_IPAR_WRITE_BAS_CONSTRAINTS, 75
 MSK_IPAR_WRITE_BAS_HEAD, 75
 MSK_IPAR_WRITE_BAS_VARIABLES, 76
 MSK_IPAR_WRITE_COMPRESSION, 76
 MSK_IPAR_WRITE_DATA_PARAM, 76
 MSK_IPAR_WRITE_FREE_CON, 76
 MSK_IPAR_WRITE_GENERIC_NAMES, 76
 MSK_IPAR_WRITE_GENERIC_NAMES_IO, 76
 MSK_IPAR_WRITE_IGNORE_INCOMPATIBLE_ITEMS, 76
 MSK_IPAR_WRITE_INT_CONSTRAINTS, 77
 MSK_IPAR_WRITE_INT_HEAD, 77
 MSK_IPAR_WRITE_INT_VARIABLES, 77
 MSK_IPAR_WRITE_LP_FULL_OBJ, 77
 MSK_IPAR_WRITE_LP_LINE_WIDTH, 77
 MSK_IPAR_WRITE_LP_QUOTED_NAMES, 77
 MSK_IPAR_WRITE_LP_STRICT_FORMAT, 77
 MSK_IPAR_WRITE_LP_TERMS_PER_LINE, 78
 MSK_IPAR_WRITE_MPS_FORMAT, 78
 MSK_IPAR_WRITE_MPS_INT, 78
 MSK_IPAR_WRITE_PRECISION, 78
 MSK_IPAR_WRITE_SOL_BARVARIABLES, 78
 MSK_IPAR_WRITE_SOL_CONSTRAINTS, 78
 MSK_IPAR_WRITE_SOL_HEAD, 78
 MSK_IPAR_WRITE_SOL_IGNORE_INVALID_NAMES, 78
 MSK_IPAR_WRITE_SOL_VARIABLES, 79
 MSK_IPAR_WRITE_TASK_INC_SOL, 79
 MSK_IPAR_WRITE_XML_MODE, 79
 String parameters, 79
 MSK_SPAR_BAS_SOL_FILE_NAME, 79
 MSK_SPAR_DATA_FILE_NAME, 79
 MSK_SPAR_DEBUG_FILE_NAME, 79
 MSK_SPAR_INT_SOL_FILE_NAME, 79
 MSK_SPAR_ITR_SOL_FILE_NAME, 79
 MSK_SPAR_MIO_DEBUG_STRING, 80
 MSK_SPAR_PARAM_COMMENT_SIGN, 80
 MSK_SPAR_PARAM_READ_FILE_NAME, 80
 MSK_SPAR_PARAM_WRITE_FILE_NAME, 80
 MSK_SPAR_READ_MPS_BOU_NAME, 80
 MSK_SPAR_READ_MPS_OBJ_NAME, 80
 MSK_SPAR_READ_MPS_RAN_NAME, 80
 MSK_SPAR_READ_MPS_RHS_NAME, 80
 MSK_SPAR_REMOTE_ACCESS_TOKEN, 81
 MSK_SPAR_SENSITIVITY_FILE_NAME, 81
 MSK_SPAR_SENSITIVITY_RES_FILE_NAME, 81
 MSK_SPAR_SOL_FILTER_XC_LOW, 81
 MSK_SPAR_SOL_FILTER_XC_UPR, 81
 MSK_SPAR_SOL_FILTER_XX_LOW, 81
 MSK_SPAR_SOL_FILTER_XX_UPR, 81
 MSK_SPAR_STAT_FILE_NAME, 82
 MSK_SPAR_STAT_KEY, 82

MSK_SPAR_STAT_NAME, 82
 MSK_SPAR_WRITE_LP_GEN_VAR_NAME, 82

Response codes

Termination, 82
 MSK_RES_OK, 82
 MSK_RES_TRM_INTERNAL, 83
 MSK_RES_TRM_INTERNAL_STOP, 83
 MSK_RES_TRM_MAX_ITERATIONS, 82
 MSK_RES_TRM_MAX_NUM_SETBACKS, 83
 MSK_RES_TRM_MAX_TIME, 82
 MSK_RES_TRM_MIO_NUM_BRANCHES, 82
 MSK_RES_TRM_MIO_NUM_RELAXS, 82
 MSK_RES_TRM_NUM_MAX_NUM_INT_SOLUTIONS, 82
 MSK_RES_TRM_NUMERICAL_PROBLEM, 83
 MSK_RES_TRM_OBJECTIVE_RANGE, 82
 MSK_RES_TRM_STALL, 83
 MSK_RES_TRM_USER_CALLBACK, 83
 Warnings, 83
 MSK_RES_WRN_ANA_ALMOST_INT_BOUNDS, 85
 MSK_RES_WRN_ANA_C_ZERO, 85
 MSK_RES_WRN_ANA_CLOSE_BOUNDS, 85
 MSK_RES_WRN_ANA_EMPTY_COLS, 85
 MSK_RES_WRN_ANA_LARGE_BOUNDS, 85
 MSK_RES_WRN_DROPPED_NZ_QOBJ, 84
 MSK_RES_WRN_DUPLICATE_BARVARIABLE_NAMES, 85
 MSK_RES_WRN_DUPLICATE_CONE_NAMES, 85
 MSK_RES_WRN_DUPLICATE_CONSTRAINT_NAMES, 85
 MSK_RES_WRN_DUPLICATE_VARIABLE_NAMES, 85
 MSK_RES_WRN_ELIMINATOR_SPACE, 85
 MSK_RES_WRN_EMPTY_NAME, 84
 MSK_RES_WRN_EXP_CONES_WITH_VARIABLES_FIXED_AT_ZERO, 85
 MSK_RES_WRN_IGNORE_INTEGER, 84
 MSK_RES_WRN_INCOMPLETE_LINEAR_DEPENDENCY_CHECK, 84
 MSK_RES_WRN_LARGE_AIJ, 83
 MSK_RES_WRN_LARGE_BOUND, 83
 MSK_RES_WRN_LARGE_CJ, 83
 MSK_RES_WRN_LARGE_CON_FX, 83
 MSK_RES_WRN_LARGE_LO_BOUND, 83
 MSK_RES_WRN_LARGE_UP_BOUND, 83
 MSK_RES_WRN_LICENSE_EXPIRE, 84
 MSK_RES_WRN_LICENSE_FEATURE_EXPIRE, 84
 MSK_RES_WRN_LICENSE_SERVER, 84
 MSK_RES_WRN_LP_DROP_VARIABLE, 84
 MSK_RES_WRN_LP_OLD_QUAD_FORMAT, 83
 MSK_RES_WRN_MIO_INFEASIBLE_FINAL, 84
 MSK_RES_WRN_MPS_SPLIT_BOU_VECTOR, 83
 MSK_RES_WRN_MPS_SPLIT_RAN_VECTOR, 83
 MSK_RES_WRN_MPS_SPLIT_RHS_VECTOR, 83
 MSK_RES_WRN_NAME_MAX_LEN, 83
 MSK_RES_WRN_NO_DUALIZER, 85
 MSK_RES_WRN_NO_GLOBAL_OPTIMIZER, 84
 MSK_RES_WRN_NZ_IN_UPR_TRI, 84
 MSK_RES_WRN_OPEN_PARAM_FILE, 83
 MSK_RES_WRN_PARAM_IGNORED_CMIO, 84
 MSK_RES_WRN_PARAM_NAME_DOU, 84

MSK_RES_WRN_PARAM_NAME_INT, 84
 MSK_RES_WRN_PARAM_NAME_STR, 84
 MSK_RES_WRN_PARAM_STR_VALUE, 84
 MSK_RES_WRN_POW_CONES_WITH_ROOT_FIXED_AT_ZERO, 85
 MSK_RES_WRN_PREOLVE_OUTOFSPACE, 85
 MSK_RES_WRN_QUAD_CONES_WITH_ROOT_FIXED_AT_ZERO, 85
 MSK_RES_WRN_RQUAD_CONES_WITH_ROOT_FIXED_AT_ZERO, 85
 MSK_RES_WRN_SOL_FILE_IGNORED_CON, 84
 MSK_RES_WRN_SOL_FILE_IGNORED_VAR, 84
 MSK_RES_WRN_SOL_FILTER, 84
 MSK_RES_WRN_SPAR_MAX_LEN, 83
 MSK_RES_WRN_SYM_MAT_LARGE, 85
 MSK_RES_WRN_TOO_FEW_BASIS_VARS, 84
 MSK_RES_WRN_TOO_MANY_BASIS_VARS, 84
 MSK_RES_WRN_UNDEF_SOL_FILE_NAME, 84
 MSK_RES_WRN_USING_GENERIC_NAMES, 84
 MSK_RES_WRN_WRITE_CHANGED_NAMES, 85
 MSK_RES_WRN_WRITE_DISCARDED_CFIX, 85
 MSK_RES_WRN_ZERO_AIJ, 83
 MSK_RES_WRN_ZEROS_IN_SPARSE_COL, 84
 MSK_RES_WRN_ZEROS_IN_SPARSE_ROW, 84
 Errors, 86
 MSK_RES_ERR_AD_INVALID_CODELIST, 97
 MSK_RES_ERR_API_ARRAY_TOO_SMALL, 96
 MSK_RES_ERR_API_CB_CONNECT, 96
 MSK_RES_ERR_API_FATAL_ERROR, 96
 MSK_RES_ERR_API_INTERNAL, 96
 MSK_RES_ERR_APPENDING_TOO_BIG_CONE, 93
 MSK_RES_ERR_ARG_IS_TOO_LARGE, 91
 MSK_RES_ERR_ARG_IS_TOO_SMALL, 90
 MSK_RES_ERR_ARGUMENT_DIMENSION, 90
 MSK_RES_ERR_ARGUMENT_IS_TOO_LARGE, 98
 MSK_RES_ERR_ARGUMENT_LENNEQ, 90
 MSK_RES_ERR_ARGUMENT_PERM_ARRAY, 93
 MSK_RES_ERR_ARGUMENT_TYPE, 90
 MSK_RES_ERR_BAR_VAR_DIM, 97
 MSK_RES_ERR_BASIS, 92
 MSK_RES_ERR_BASIS_FACTOR, 95
 MSK_RES_ERR_BASIS_SINGULAR, 95
 MSK_RES_ERR_BLANK_NAME, 87
 MSK_RES_ERR_CBF_DUPLICATE_ACOORD, 99
 MSK_RES_ERR_CBF_DUPLICATE_BCOORD, 99
 MSK_RES_ERR_CBF_DUPLICATE_CON, 99
 MSK_RES_ERR_CBF_DUPLICATE_INT, 99
 MSK_RES_ERR_CBF_DUPLICATE_OBJ, 98
 MSK_RES_ERR_CBF_DUPLICATE_OBJACCOORD, 99
 MSK_RES_ERR_CBF_DUPLICATE_POW_CONES, 99
 MSK_RES_ERR_CBF_DUPLICATE_POW_STAR_CONES, 99
 MSK_RES_ERR_CBF_DUPLICATE_PSDVAR, 99
 MSK_RES_ERR_CBF_DUPLICATE_VAR, 99
 MSK_RES_ERR_CBF_INVALID_CON_TYPE, 99
 MSK_RES_ERR_CBF_INVALID_DIMENSION_OF_CONES, 100
 MSK_RES_ERR_CBF_INVALID_DOMAIN_DIMENSION, 99
 MSK_RES_ERR_CBF_INVALID_EXP_DIMENSION, 99
 MSK_RES_ERR_CBF_INVALID_INT_INDEX, 99
 MSK_RES_ERR_CBF_INVALID_NUMBER_OF_CONES, 100
 MSK_RES_ERR_CBF_INVALID_POWER, 99
 MSK_RES_ERR_CBF_INVALID_POWER_CONE_INDEX, 99
 MSK_RES_ERR_CBF_INVALID_POWER_STAR_CONE_INDEX, 99
 MSK_RES_ERR_CBF_INVALID_PSDVAR_DIMENSION, 99
 MSK_RES_ERR_CBF_INVALID_VAR_TYPE, 99
 MSK_RES_ERR_CBF_NO_VARIABLES, 98
 MSK_RES_ERR_CBF_NO_VERSION_SPECIFIED, 98
 MSK_RES_ERR_CBF_OBJ_SENSE, 98
 MSK_RES_ERR_CBF_PARSE, 98
 MSK_RES_ERR_CBF_POWER_CONE_IS_TOO_LONG, 99
 MSK_RES_ERR_CBF_POWER_CONE_MISMATCH, 99
 MSK_RES_ERR_CBF_POWER_STAR_CONE_MISMATCH, 99
 MSK_RES_ERR_CBF_SYNTAX, 98
 MSK_RES_ERR_CBF_TOO_FEW_CONSTRAINTS, 99
 MSK_RES_ERR_CBF_TOO_FEW_INTS, 99
 MSK_RES_ERR_CBF_TOO_FEW_PSDVAR, 99
 MSK_RES_ERR_CBF_TOO_FEW_VARIABLES, 99
 MSK_RES_ERR_CBF_TOO_MANY_CONSTRAINTS, 98
 MSK_RES_ERR_CBF_TOO_MANY_INTS, 99
 MSK_RES_ERR_CBF_TOO_MANY_VARIABLES, 98
 MSK_RES_ERR_CBF_UNHANDLED_POWER_CONE_TYPE, 99
 MSK_RES_ERR_CBF_UNHANDLED_POWER_STAR_CONE_TYPE, 99
 MSK_RES_ERR_CBF_UNSUPPORTED, 99
 MSK_RES_ERR_CON_Q_NOT_NSD, 93
 MSK_RES_ERR_CON_Q_NOT_PSD, 92
 MSK_RES_ERR_CONE_INDEX, 93
 MSK_RES_ERR_CONE_OVERLAP, 93
 MSK_RES_ERR_CONE_OVERLAP_APPEND, 93
 MSK_RES_ERR_CONE_PARAMETER, 93
 MSK_RES_ERR_CONE_REP_VAR, 93
 MSK_RES_ERR_CONE_SIZE, 93
 MSK_RES_ERR_CONE_TYPE, 93
 MSK_RES_ERR_CONE_TYPE_STR, 93
 MSK_RES_ERR_DATA_FILE_EXT, 87
 MSK_RES_ERR_DUP_NAME, 87
 MSK_RES_ERR_DUPLICATE_AIJ, 93
 MSK_RES_ERR_DUPLICATE_BARVARIABLE_NAMES, 98
 MSK_RES_ERR_DUPLICATE_CONE_NAMES, 98
 MSK_RES_ERR_DUPLICATE_CONSTRAINT_NAMES, 97
 MSK_RES_ERR_DUPLICATE_VARIABLE_NAMES, 97
 MSK_RES_ERR_END_OF_FILE, 87
 MSK_RES_ERR_FACTOR, 95
 MSK_RES_ERR_FEASREPAIR_CANNOT_RELAX, 95
 MSK_RES_ERR_FEASREPAIR_INCONSISTENT_BOUND, 95
 MSK_RES_ERR_FEASREPAIR_SOLVING_RELAXED, 95

MSK_RES_ERR_FILE_LICENSE, 86
 MSK_RES_ERR_FILE_OPEN, 87
 MSK_RES_ERR_FILE_READ, 87
 MSK_RES_ERR_FILE_WRITE, 87
 MSK_RES_ERR_FINAL_SOLUTION, 95
 MSK_RES_ERR_FIRST, 95
 MSK_RES_ERR_FIRSTI, 92
 MSK_RES_ERR_FIRSTJ, 92
 MSK_RES_ERR_FIXED_BOUND_VALUES, 94
 MSK_RES_ERR_FLEXLM, 86
 MSK_RES_ERR_FORMAT_STRING, 87
 MSK_RES_ERR_GLOBAL_INV_CONIC_PROBLEM, 95
 MSK_RES_ERR_HUGE_AIJ, 93
 MSK_RES_ERR_HUGE_C, 93
 MSK_RES_ERR_IDENTICAL_TASKS, 97
 MSK_RES_ERR_IN_ARGUMENT, 90
 MSK_RES_ERR_INDEX, 91
 MSK_RES_ERR_INDEX_ARR_IS_TOO_LARGE, 90
 MSK_RES_ERR_INDEX_ARR_IS_TOO_SMALL, 90
 MSK_RES_ERR_INDEX_IS_TOO_LARGE, 90
 MSK_RES_ERR_INDEX_IS_TOO_SMALL, 90
 MSK_RES_ERR_INF_DOU_INDEX, 90
 MSK_RES_ERR_INF_DOU_NAME, 91
 MSK_RES_ERR_INF_INT_INDEX, 90
 MSK_RES_ERR_INF_INT_NAME, 91
 MSK_RES_ERR_INF_LINT_INDEX, 90
 MSK_RES_ERR_INF_LINT_NAME, 91
 MSK_RES_ERR_INF_TYPE, 91
 MSK_RES_ERR_INFEAS_UNDEFINED, 97
 MSK_RES_ERR_INFINITE_BOUND, 93
 MSK_RES_ERR_INT64_TO_INT32_CAST, 97
 MSK_RES_ERR_INTERNAL, 96
 MSK_RES_ERR_INTERNAL_TEST_FAILED, 97
 MSK_RES_ERR_INV_APTRE, 91
 MSK_RES_ERR_INV_BK, 91
 MSK_RES_ERR_INV_BKC, 91
 MSK_RES_ERR_INV_BKX, 91
 MSK_RES_ERR_INV_CONE_TYPE, 92
 MSK_RES_ERR_INV_CONE_TYPE_STR, 92
 MSK_RES_ERR_INV_MARKI, 96
 MSK_RES_ERR_INV_MARKJ, 96
 MSK_RES_ERR_INV_NAME_ITEM, 92
 MSK_RES_ERR_INV_NUMI, 96
 MSK_RES_ERR_INV_NUMJ, 96
 MSK_RES_ERR_INV_OPTIMIZER, 95
 MSK_RES_ERR_INV_PROBLEM, 94
 MSK_RES_ERR_INV_QCON_SUBI, 94
 MSK_RES_ERR_INV_QCON_SUBJ, 94
 MSK_RES_ERR_INV_QCON_SUBK, 94
 MSK_RES_ERR_INV_QCON_VAL, 94
 MSK_RES_ERR_INV_QOBJ_SUBI, 94
 MSK_RES_ERR_INV_QOBJ_SUBJ, 94
 MSK_RES_ERR_INV_QOBJ_VAL, 94
 MSK_RES_ERR_INV_SK, 92
 MSK_RES_ERR_INV_SK_STR, 92
 MSK_RES_ERR_INV_SKC, 92
 MSK_RES_ERR_INV_SKN, 92
 MSK_RES_ERR_INV_SKX, 92
 MSK_RES_ERR_INV_VAR_TYPE, 91
 MSK_RES_ERR_INVALID_AIJ, 94
 MSK_RES_ERR_INVALID_AMPL_STUB, 97
 MSK_RES_ERR_INVALID_BARVAR_NAME, 88
 MSK_RES_ERR_INVALID_CJ, 94
 MSK_RES_ERR_INVALID_COMPRESSION, 95
 MSK_RES_ERR_INVALID_CON_NAME, 88
 MSK_RES_ERR_INVALID_CONE_NAME, 88
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CFIX, 97
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_CONES, 97
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_FREE_CONSTRAINTS, 97
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_NONLINEAR, 97
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_RANGED_CONSTRAINTS, 97
 MSK_RES_ERR_INVALID_FILE_FORMAT_FOR_SYM_MAT, 97
 MSK_RES_ERR_INVALID_FILE_NAME, 87
 MSK_RES_ERR_INVALID_FORMAT_TYPE, 92
 MSK_RES_ERR_INVALID_IDX, 91
 MSK_RES_ERR_INVALID_IOMODE, 95
 MSK_RES_ERR_INVALID_MAX_NUM, 91
 MSK_RES_ERR_INVALID_NAME_IN_SOL_FILE, 89
 MSK_RES_ERR_INVALID_OBJ_NAME, 87
 MSK_RES_ERR_INVALID_OBJECTIVE_SENSE, 94
 MSK_RES_ERR_INVALID_PROBLEM_TYPE, 98
 MSK_RES_ERR_INVALID_SOL_FILE_NAME, 87
 MSK_RES_ERR_INVALID_STREAM, 87
 MSK_RES_ERR_INVALID_SURPLUS, 92
 MSK_RES_ERR_INVALID_SYM_MAT_DIM, 97
 MSK_RES_ERR_INVALID_TASK, 87
 MSK_RES_ERR_INVALID_UTF8, 96
 MSK_RES_ERR_INVALID_VAR_NAME, 88
 MSK_RES_ERR_INVALID_WCHAR, 96
 MSK_RES_ERR_INVALID_WHICH_SOL, 91
 MSK_RES_ERR_JSON_DATA, 90
 MSK_RES_ERR_JSON_FORMAT, 90
 MSK_RES_ERR_JSON_MISSING_DATA, 90
 MSK_RES_ERR_JSON_NUMBER_OVERFLOW, 90
 MSK_RES_ERR_JSON_STRING, 90
 MSK_RES_ERR_JSON_SYNTAX, 90
 MSK_RES_ERR_LAST, 95
 MSK_RES_ERR_LASTI, 92
 MSK_RES_ERR_LASTJ, 92
 MSK_RES_ERR_LAU_ARG_K, 98
 MSK_RES_ERR_LAU_ARG_M, 98
 MSK_RES_ERR_LAU_ARG_N, 98
 MSK_RES_ERR_LAU_ARG_TRANS, 98
 MSK_RES_ERR_LAU_ARG_TRANSA, 98
 MSK_RES_ERR_LAU_ARG_TRANSB, 98
 MSK_RES_ERR_LAU_ARG_UPLO, 98
 MSK_RES_ERR_LAU_INVALID_LOWER_TRIANGULAR_MATRIX, 98
 MSK_RES_ERR_LAU_INVALID_SPARSE_SYMMETRIC_MATRIX, 98

MSK_RES_ERR_LAU_NOT_POSITIVE_DEFINITE, 98
 MSK_RES_ERR_LAU_SINGULAR_MATRIX, 98
 MSK_RES_ERR_LAU_UNKNOWN, 98
 MSK_RES_ERR_LICENSE, 86
 MSK_RES_ERR_LICENSE_CANNOT_ALLOCATE, 86
 MSK_RES_ERR_LICENSE_CANNOT_CONNECT, 86
 MSK_RES_ERR_LICENSE_EXPIRED, 86
 MSK_RES_ERR_LICENSE_FEATURE, 86
 MSK_RES_ERR_LICENSE_INVALID_HOSTID, 86
 MSK_RES_ERR_LICENSE_MAX, 86
 MSK_RES_ERR_LICENSE_MOSEKLM_DAEMON, 86
 MSK_RES_ERR_LICENSE_NO_SERVER_LINE, 86
 MSK_RES_ERR_LICENSE_NO_SERVER_SUPPORT, 86
 MSK_RES_ERR_LICENSE_SERVER, 86
 MSK_RES_ERR_LICENSE_SERVER_VERSION, 86
 MSK_RES_ERR_LICENSE_VERSION, 86
 MSK_RES_ERR_LINK_FILE_DLL, 87
 MSK_RES_ERR_LIVING_TASKS, 87
 MSK_RES_ERR_LOWER_BOUND_IS_A_NAN, 93
 MSK_RES_ERR_LP_DUP_SLACK_NAME, 89
 MSK_RES_ERR_LP_EMPTY, 89
 MSK_RES_ERR_LP_FILE_FORMAT, 89
 MSK_RES_ERR_LP_FORMAT, 89
 MSK_RES_ERR_LP_FREE_CONSTRAINT, 89
 MSK_RES_ERR_LP_INCOMPATIBLE, 89
 MSK_RES_ERR_LP_INVALID_CON_NAME, 89
 MSK_RES_ERR_LP_INVALID_VAR_NAME, 89
 MSK_RES_ERR_LP_WRITE_CONIC_PROBLEM, 89
 MSK_RES_ERR_LP_WRITE_GECO_PROBLEM, 89
 MSK_RES_ERR_LU_MAX_NUM_TRIES, 96
 MSK_RES_ERR_MAX_LEN_IS_TOO_SMALL, 92
 MSK_RES_ERR_MAXNUMBARVAR, 91
 MSK_RES_ERR_MAXNUMCON, 91
 MSK_RES_ERR_MAXNUMCONE, 93
 MSK_RES_ERR_MAXNUMQNZ, 91
 MSK_RES_ERR_MAXNUMVAR, 91
 MSK_RES_ERR_MIO_INTERNAL, 98
 MSK_RES_ERR_MIO_INVALID_NODE_OPTIMIZER, 100
 MSK_RES_ERR_MIO_INVALID_ROOT_OPTIMIZER, 100
 MSK_RES_ERR_MIO_NO_OPTIMIZER, 95
 MSK_RES_ERR_MISSING_LICENSE_FILE, 86
 MSK_RES_ERR_MIXED_CONIC_AND_NL, 95
 MSK_RES_ERR_MPS_CONE_OVERLAP, 88
 MSK_RES_ERR_MPS_CONE_REPEAT, 88
 MSK_RES_ERR_MPS_CONE_TYPE, 88
 MSK_RES_ERR_MPS_DUPLICATE_Q_ELEMENT, 89
 MSK_RES_ERR_MPS_FILE, 88
 MSK_RES_ERR_MPS_INV_BOUND_KEY, 88
 MSK_RES_ERR_MPS_INV_CON_KEY, 88
 MSK_RES_ERR_MPS_INV_FIELD, 88
 MSK_RES_ERR_MPS_INV_MARKER, 88
 MSK_RES_ERR_MPS_INV_SEC_NAME, 88
 MSK_RES_ERR_MPS_INV_SEC_ORDER, 88
 MSK_RES_ERR_MPS_INVALID_OBJ_NAME, 89
 MSK_RES_ERR_MPS_INVALID_OBJSENSE, 89
 MSK_RES_ERR_MPS_MUL_CON_NAME, 88
 MSK_RES_ERR_MPS_MUL_CSEC, 88
 MSK_RES_ERR_MPS_MUL_QOBJ, 88
 MSK_RES_ERR_MPS_MUL_QSEC, 88
 MSK_RES_ERR_MPS_NO_OBJECTIVE, 88
 MSK_RES_ERR_MPS_NON_SYMMETRIC_Q, 88
 MSK_RES_ERR_MPS_NULL_CON_NAME, 88
 MSK_RES_ERR_MPS_NULL_VAR_NAME, 88
 MSK_RES_ERR_MPS_SPLITTED_VAR, 88
 MSK_RES_ERR_MPS_TAB_IN_FIELD2, 89
 MSK_RES_ERR_MPS_TAB_IN_FIELD3, 89
 MSK_RES_ERR_MPS_TAB_IN_FIELD5, 89
 MSK_RES_ERR_MPS_UNDEF_CON_NAME, 88
 MSK_RES_ERR_MPS_UNDEF_VAR_NAME, 88
 MSK_RES_ERR_MUL_A_ELEMENT, 91
 MSK_RES_ERR_NAME_IS_NULL, 95
 MSK_RES_ERR_NAME_MAX_LEN, 95
 MSK_RES_ERR_NAN_IN_BLC, 94
 MSK_RES_ERR_NAN_IN_BLX, 94
 MSK_RES_ERR_NAN_IN_BUC, 94
 MSK_RES_ERR_NAN_IN_BUX, 94
 MSK_RES_ERR_NAN_IN_C, 94
 MSK_RES_ERR_NAN_IN_DOUBLE_DATA, 94
 MSK_RES_ERR_NEGATIVE_APPEND, 95
 MSK_RES_ERR_NEGATIVE_SURPLUS, 95
 MSK_RES_ERR_NEWER_DLL, 87
 MSK_RES_ERR_NO_BARS_FOR_SOLUTION, 97
 MSK_RES_ERR_NO_BARX_FOR_SOLUTION, 97
 MSK_RES_ERR_NO_BASIS_SOL, 95
 MSK_RES_ERR_NO_DUAL_FOR_ITG_SOL, 96
 MSK_RES_ERR_NO_DUAL_INFEAS_CER, 95
 MSK_RES_ERR_NO_INIT_ENV, 87
 MSK_RES_ERR_NO_OPTIMIZER_VAR_TYPE, 95
 MSK_RES_ERR_NO_PRIMAL_INFEAS_CER, 95
 MSK_RES_ERR_NO_SNX_FOR_BAS_SOL, 96
 MSK_RES_ERR_NO_SOLUTION_IN_CALLBACK, 96
 MSK_RES_ERR_NON_UNIQUE_ARRAY, 98
 MSK_RES_ERR_NONCONVEX, 92
 MSK_RES_ERR_NONLINEAR_EQUALITY, 92
 MSK_RES_ERR_NONLINEAR_RANGED, 92
 MSK_RES_ERR_NULL_ENV, 87
 MSK_RES_ERR_NULL_POINTER, 87
 MSK_RES_ERR_NULL_TASK, 87
 MSK_RES_ERR_NUM_ARGUMENTS, 90
 MSK_RES_ERR_NUMCONLIM, 91
 MSK_RES_ERR_NUMVARLIM, 91
 MSK_RES_ERR_OBJ_Q_NOT_NSD, 93
 MSK_RES_ERR_OBJ_Q_NOT_PSD, 93
 MSK_RES_ERR_OBJECTIVE_RANGE, 92
 MSK_RES_ERR_OLDER_DLL, 86
 MSK_RES_ERR_OPF_FORMAT, 89
 MSK_RES_ERR_OPF_NEW_VARIABLE, 89
 MSK_RES_ERR_OPF_PREMATURE_EOF, 89
 MSK_RES_ERR_OPTIMIZER_LICENSE, 86
 MSK_RES_ERR_OVERFLOW, 95
 MSK_RES_ERR_PARAM_INDEX, 90
 MSK_RES_ERR_PARAM_IS_TOO_LARGE, 90
 MSK_RES_ERR_PARAM_IS_TOO_SMALL, 90
 MSK_RES_ERR_PARAM_NAME, 90
 MSK_RES_ERR_PARAM_NAME_DOU, 90
 MSK_RES_ERR_PARAM_NAME_INT, 90

MSK_RES_ERR_PARAM_NAME_STR, 90
 MSK_RES_ERR_PARAM_TYPE, 90
 MSK_RES_ERR_PARAM_VALUE_STR, 90
 MSK_RES_ERR_PLATFORM_NOT_LICENSED, 86
 MSK_RES_ERR_POSTSOLVE, 95
 MSK_RES_ERR_PRO_ITEM, 92
 MSK_RES_ERR_PROB_LICENSE, 86
 MSK_RES_ERR_PTF_FORMAT, 89
 MSK_RES_ERR_QCON_SUBI_TOO_LARGE, 94
 MSK_RES_ERR_QCON_SUBI_TOO_SMALL, 94
 MSK_RES_ERR_QCON_UPPER_TRIANGLE, 94
 MSK_RES_ERR_QOBJ_UPPER_TRIANGLE, 94
 MSK_RES_ERR_READ_FORMAT, 88
 MSK_RES_ERR_READ_LP_MISSING_END_TAG, 89
 MSK_RES_ERR_READ_LP_NONEXISTING_NAME, 89
 MSK_RES_ERR_REMOVE_CONE_VARIABLE, 93
 MSK_RES_ERR_REPAIR_INVALID_PROBLEM, 95
 MSK_RES_ERR_REPAIR_OPTIMIZATION_FAILED, 95
 MSK_RES_ERR_SEN_BOUND_INVALID_LO, 96
 MSK_RES_ERR_SEN_BOUND_INVALID_UP, 96
 MSK_RES_ERR_SEN_FORMAT, 96
 MSK_RES_ERR_SEN_INDEX_INVALID, 96
 MSK_RES_ERR_SEN_INDEX_RANGE, 96
 MSK_RES_ERR_SEN_INVALID_REGEX, 96
 MSK_RES_ERR_SEN_NUMERICAL, 96
 MSK_RES_ERR_SEN_SOLUTION_STATUS, 96
 MSK_RES_ERR_SEN_UNDEF_NAME, 96
 MSK_RES_ERR_SEN_UNHANDLED_PROBLEM_TYPE, 96
 MSK_RES_ERR_SERVER_CONNECT, 100
 MSK_RES_ERR_SERVER_PROBLEM_SIZE, 100
 MSK_RES_ERR_SERVER_PROTOCOL, 100
 MSK_RES_ERR_SERVER_STATUS, 100
 MSK_RES_ERR_SERVER_TOKEN, 100
 MSK_RES_ERR_SHAPE_IS_TOO_LARGE, 90
 MSK_RES_ERR_SIZE_LICENSE, 86
 MSK_RES_ERR_SIZE_LICENSE_CON, 86
 MSK_RES_ERR_SIZE_LICENSE_INTVAR, 86
 MSK_RES_ERR_SIZE_LICENSE_NUMCORES, 97
 MSK_RES_ERR_SIZE_LICENSE_VAR, 86
 MSK_RES_ERR_SLICE_SIZE, 95
 MSK_RES_ERR_SOL_FILE_INVALID_NUMBER, 93
 MSK_RES_ERR_SOLITEM, 91
 MSK_RES_ERR_SOLVER_PROBTYPE, 92
 MSK_RES_ERR_SPACE, 87
 MSK_RES_ERR_SPACE_LEAKING, 88
 MSK_RES_ERR_SPACE_NO_INFO, 88
 MSK_RES_ERR_SYM_MAT_DUPLICATE, 97
 MSK_RES_ERR_SYM_MAT_HUGE, 94
 MSK_RES_ERR_SYM_MAT_INVALID, 94
 MSK_RES_ERR_SYM_MAT_INVALID_COL_INDEX, 97
 MSK_RES_ERR_SYM_MAT_INVALID_ROW_INDEX, 97
 MSK_RES_ERR_SYM_MAT_INVALID_VALUE, 97
 MSK_RES_ERR_SYM_MAT_NOT_LOWER_TRINGULAR, 97
 MSK_RES_ERR_TASK_INCOMPATIBLE, 96
 MSK_RES_ERR_TASK_INVALID, 96
 MSK_RES_ERR_TASK_WRITE, 96
 MSK_RES_ERR_THREAD_COND_INIT, 87
 MSK_RES_ERR_THREAD_CREATE, 87
 MSK_RES_ERR_THREAD_MUTEX_INIT, 87
 MSK_RES_ERR_THREAD_MUTEX_LOCK, 87
 MSK_RES_ERR_THREAD_MUTEX_UNLOCK, 87
 MSK_RES_ERR_TOCONIC_CONSTR_NOT_CONIC, 100
 MSK_RES_ERR_TOCONIC_CONSTR_Q_NOT_PSD, 100
 MSK_RES_ERR_TOCONIC_CONSTRAINT_FX, 100
 MSK_RES_ERR_TOCONIC_CONSTRAINT_RA, 100
 MSK_RES_ERR_TOCONIC_OBJECTIVE_NOT_PSD, 100
 MSK_RES_ERR_TOO_SMALL_A_TRUNCATION_VALUE, 94
 MSK_RES_ERR_TOO_SMALL_MAX_NUM_NZ, 91
 MSK_RES_ERR_TOO_SMALL_MAXNUMANZ, 91
 MSK_RES_ERR_UNB_STEP_SIZE, 97
 MSK_RES_ERR_UNDEF_SOLUTION, 92
 MSK_RES_ERR_UNDEFINED_OBJECTIVE_SENSE, 94
 MSK_RES_ERR_UNHANDLED_SOLUTION_STATUS, 98
 MSK_RES_ERR_UNKNOWN, 87
 MSK_RES_ERR_UPPER_BOUND_IS_A_NAN, 93
 MSK_RES_ERR_UPPER_TRIANGLE, 98
 MSK_RES_ERR_WHICHITEM_NOT_ALLOWED, 91
 MSK_RES_ERR_WHICHSOL, 91
 MSK_RES_ERR_WRITE_LP_FORMAT, 89
 MSK_RES_ERR_WRITE_LP_NON_UNIQUE_NAME, 89
 MSK_RES_ERR_WRITE_MPS_INVALID_NAME, 89
 MSK_RES_ERR_WRITE_OPF_INVALID_VAR_NAME, 89
 MSK_RES_ERR_WRITING_FILE, 89
 MSK_RES_ERR_XML_INVALID_PROBLEM_TYPE, 97
 MSK_RES_ERR_Y_IS_UNDEFINED, 94

Index

C

CBF format, [151](#)

F

format

- CBF, [151](#)
- json, [171](#)
- LP, [125](#)
- MPS, [130](#)
- OPF, [142](#)
- PTF, [166](#)
- sol, [179](#)
- task, [171](#)

J

json format, [171](#)

L

license, [15](#)

LP format, [125](#)

M

MPS format, [130](#)
free, [141](#)

O

OPF format, [142](#)

P

PTF format, [166](#)

S

sol format, [179](#)

solution

- file format, [179](#)

T

task format, [171](#)