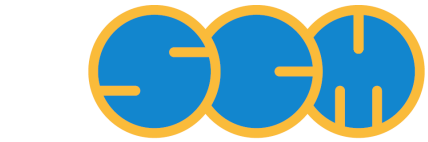

Installation Manual

ADF Program System
Release 2013

Scientific Computing & Modelling NV
Vrije Universiteit, Theoretical Chemistry
De Boelelaan 1083; 1081 HV Amsterdam; The Netherlands
WWW: www.scm.com
E-mail: support@scm.com

Table of Contents

1. Introduction

The Installation Manual describes installation of the ADF program package on the platforms on which it is supported. This document applies to all systems on which ADF is supported, that is including Windows and Mac OS X. For optimal performance, some system specific tuning may be needed. Therefore, it is strongly recommended to also read Chapter 4 of this document. This Installation Manual is available from the Documentation part of the SCM web site:

http://www.scm.com/Doc/Welcome.html

The ADF package consists of the following main classes of programs:

- Computational engines: ADF, BAND, COSMO-RS, DFTB, and ReaxFF. Each of the engines has its own (text-based) input and output and can be used as a standalone program from scripts and the command line. Within this manual we will use the word *ADF* for all computational engines.
- Utilities and property programs. These are used primarily for pre- and post-processing data of the computational engines.
- Graphical user interface, which is used to prepare input for computational engines and to visually present their results.

You will always install the complete ADF package at once and your license file will determine which of the functionality may be used.

The whole ADF package is written with the Unix-like environment in mind. Some knowledge of Unix may or may not be required depending on the operating system on which you are going to use ADF. It also depends on whether you are going to use ADF via the GUI only or also from the command line. For example, if you are going to use ADF from within the GUI on Windows or Mac OS X, then you do not need to know anything about Unix. If you are going to use ADF from the command line then you will need to know how to write shell scripts and have some knowledge of the environment variables. If you the one who is going to install the package on a Unix-like system, such as Linux, then you need to know how to modify shell resource files such as .bashrc.

1.1 Requirements

Hardware and software requirements for the ADF package depend on the platform. The list of supported platforms, including information on the operating system, parallel environment (MPI), and compilers used is available in the Download section of our web site:

http://www.scm.com/Downloads/Welcome.html

Hardware requirements

Memory

In a parallel calculation, the total amount of memory used by the job is a sum of that used by each process. Starting from ADF2010, some large chunks of data are placed in the shared memory so the sum rule does not strictly hold. In principle, it is more correct to count memory per process but since ADF is an MPI program it runs most efficiently when the number of processes corresponds to the number of physical processors cores. Therefore, all memory amounts below are per processor core.

The amount of RAM per core needed to run ADF depends greatly on the kind of calculation you perform. For small calculations, 256 MB will be sufficient, but if there is a lot of memory available ADF may run significantly faster. A large amount memory also reduces the load on the disks, which may speed up your calculation depending on the I/O sub-system and the operating system.

The memory requirement increases with the system size. For example, for a molecule containing 100 atoms with a DZ basis set it may be sufficient to have 256 MB but for a molecule with 1000 atoms up to 8 gigabytes may be required. Also, if you are going to perform TDDFT, relativistic spin-orbit or analytical frequency calculations then the amount of memory should be larger. As an indication, an analytical vibrational frequency calculation of a organometallic complex containing 105 atoms with a TZP basis set uses up to 1GB of RAM per process but it can be done with less, even though not as efficiently.

Disk

For installation of the package on Linux/Unix you need from about 1GB (without sources) to 3GB (with sources and compiled objects). The run-time (scratch) disk space requirements greatly depend on what type of calculation you perform. For the ADF program, it may range from a few megabytes for a small molecule up to a hundred gigabytes for a large molecule with a large basis set, the amount scaling quadratically with the system size. BAND calculations typically require a lot more disk space than ADF for a system of a similar size. You may need anywhere from a few to hundreds of gigabytes of free disk space at run time.

Network

First of all, on the most popular systems (Linux, Mac OS and Windows) a network card must be present in the computer as its hardware MAC address is used as the computer's ID for the licensing. Besides that, when using Platform-MPI, the computer must be connected to a (possibly virtual) network and have a valid IP address. The IP address must not change while an MPI job is running. The latter is mostly relevant for laptop users.

In order to enable MPI on a standalone Windows computer, one may need to create a dummy network connection by adding a network "card" called Microsoft Loopback Adapter. This interface will be assigned an IP address from a private range.

Multi-host parallel performance. As far as performance concerned, a switched Gigabit Ethernet network is typically sufficient for good parallel performance on up to four nodes. If you are going to use more nodes you may need faster communication hardware, such as Infiniband, to get good performance. Please note that multi-host execution is not supported on Windows.

Software requirements

Operating System

The package runs on Windows and on the following Unix variants: Linux (i686, x86-64, ia64, PowerPC), IBM AIX, Mac OS X.

On the Apple systems, the 32-bit ADF version is supported on Mac OS X 10.5 and 10.6. The 64-bit ADF version is supported on Mac OS X 10.6 and newer.

The Windows version of ADF is supported on Windows 7, Vista, and XP with Service Pack 3.

Additional libraries

Certain version of ADF will require different libraries to be installed on the system depending on the MPI library used. For Platform-MPI on Linux, the libxml2 library must be installed.

Graphics

In order to run the the graphical user interface (GUI) the computer needs to have an OpenGL-capable graphics subsystem (hardware, drivers and libraries). Besides that, on Linux the following (or equivalent) packages must be installed:

```
fontconfig-2.4.1
freetype-2.2.1
libdrm-2.0.2
libICE-1.0.1
libSM-1.0.1
libstdc++-4.1.2
libX11-1.0.3
libXau-1.0.1
libXdmcp-1.0.1
libXext-1.0.1
libXft-2.1.10
libXrender-0.9.1
libXScrnSaver-1.1.0 (Ubuntu users may need to install libXss1)
libXt-1.0.2
libXxf86vm-1.0.1
mesa-libGL-6.5.1
mesa-libGLU-6.5.1
```

The GUI will not able to start without shared libraries provided by these packages.

Compiler

If you have a license for the source code, you can compile the source yourself, with or without your own modifications.

The source consists mainly of Fortran95 code, with some small parts written in C. Some of the Fortran2003 features are also used so a compiler supporting it is required. You must use object-compatible Fortran and C compilers to those we are using on the same platform, since some parts of the code are available only as object modules. For all x86 platforms it is currently Intel Fortran 12.1, for PowerPC it is IBM XL Fortran 12.1. It is very unlikely that other compilers, or even a different major version of the same compiler, will work with these object modules. We cannot support compilers different from what we are using ourselves.

To check which compiler to use, check the detailed machine information on the Download section of our web site.

2. Installation

Typically installation of the ADF package is simple and straightforward. If you have problems installing it, contact us for assistance at support@scm.com.

To install the ADF package you have to go through the following steps:

Below we discuss each step separately.

2.1 Decide which version to install

Choose the released version or a snapshot. Normally, you will install the released version from the main download page. The ADF2013 snapshots page contains the most recent bug fixes. You may want to install a snapshot version if you know that some bugs have been fixed recently. The development snapshots page contains the latest developments. You will only want to download it if you need to use the latest functionality not available in ADF2013.

Choose a platform. ADF is available for a number of platforms. A platform is a combination of the processor architecture, operating system, MPI implementation, and the "bitness" of the address space: 32 or 64. Currently, the following platforms are officially supported and available from our website:

- Linux:
  - i686 (32-bit): Platform-MPI
  - x86-64 (64-bit): Platform-MPI, IntelMPI, SGI MPT, Cray MPI
  - ppc64 (64-bit): IBM POE
- Mac OS X Snow Leopard and later (10.6+, including Lion): x86-64 (64-bit)
- Windows: Platform-MPI on both x86 (32-bit) and x64 (64-bit)
- Itanium2 (ia64) is not supported because there is no suitable Fortran2003 compiler exists for this architecture

32 vs. 64 bits. Generally, if your processor and operating system support 64-bit addressing, you should install the 64-bit version. It allows to run larger calculations than a 32-bit version. There is generally no or very small speed difference between 32- and 64-bit versions. The 32-bit version should only be installed on 32-bit operating systems. For Mac OS only a 64-bit version is available.

Choose MPI (on Linux). For the Linux-on-PowerPC users the choice is easy, they have to use an IBM POE version of ADF. For Linux systems running on AMD Opteron or Intel Xeon processors the choice depends on which MPI implementations can run on your hardware. Users of Cray or SGI should download a version specifically built for their platform, Cray MPI or SGI MPT, respectively.

Platform-MPI vs. IntelMPI: The choice depends on the cluster interconnect used on your system and the batch system. In general we recommend installing the IntelMPI version on system with Ethernet or Infiniband interconnect with uDAPL library. If uDAPL is not available but IBverbs or PSM is, then the recommended MPI is Platform-MPI. Users with a SLURM batch system should always install an IntelMPI version. The Platform-MPI or IntelMPI runtime environment is distributed with the corresponding ADF version.

2.2 Download and install the software

All systems: Download an installation package from one of the download pages mentioned above.

Unix and Linux: The installation package is a compressed tar archive. Untar it in the directory of your choice. Please note that in a cluster environment ADF must be installed on a shared file system accessible from all nodes of the cluster. The archive contains one directory, adf2013.01, which, when installed, will be referred to as *$ADFHOME*.

Mac OS X: The installation package is a disk image (.dmg) file. To install ADF on Mac OS X, double click on the downloaded disk image and drag *ADF2013.01* somewhere on your hard disk, such as the *Applications* directory. Be sure to **read the enclosed ReadMe** file for further important details!

Windows: The downloaded file is an executable Microsoft Installer package containing an installation wizard that will guide you through the installation process. Open the file after downloading. Please note that you need Administrator privileges to install ADF.

2.3 Set up the environment

Windows users can skip to the next section since for them the environment is modified by the installation wizard.

Mac users may follow the UNIX instructions if they (also) wish to run from the command line. However, read the ReadMe file inside the dmg file for some important extra details! Alternatively, Mac users can start by double clicking the ADF2013.01 application. The ADF2013.01 application will automatically set the environment, and start ADFjobs for you. Next, all tasks (GUI or compuational engines) started from this ADFjobs will inherit the proper environment. In older distributions ADFLaunch was used, it has been replaced by the ADF2013.01 application.

For users of any **Unix-like** system the following step is mandatory.

For a **single-user** installation, the end-user is likely also the person installing ADF. In this case it should be sufficient to edit the $ADFHOME/adfrc.sh (for bash users) or $ADFHOME/adfrc.csh (for tcsh users) file to set a number of important environment variables and source the file in your shell resource file (for bash):

```
. $HOME/adf2013.01/adfrc.sh
```

or (for tcsh)

```
source $HOME/adf2013.01/adfrc.csh
```

For a **multi-user** installation, you can either copy both adfrc.* files to the /etc/profile.d directory (after editing them) or, if your system supports modules, create a module file for ADF2013. The following environment variables must be defined in the module file:

- ADFHOME: ADF installation directory
- ADFBIN: should be equal to $ADFHOME/bin
- ADFRESOURCES: should be equal to $ADFHOME/atomicdata
- SCMLICENSE: complete path of the license file, typically $ADFHOME/license.txt
- SCM_TMPDIR: path to the user's scratch directory, for example, /scratch/$USER. This directory must exist prior to the first execution of any program from the ADF package by that user. Thus it may be a good idea to add to the user's profile a command that creates the directory in case it does not exist. You can also choose to use the same directory for all users. See Section 2.5 for more details.

A complete list of environment variables is provided in Appendix A.

2.4 Set up the license

Which functionality can be used on each computer is determined by the license file pointed to by the SCMLICENSE environment variable. If you are a **trial** user, you should receive a demo license file together with your download login and password. The default location of the license file (for Window, Mac and the default SCMLICENSE value from a adfrc.* file) is set to $ADFHOME/license.txt, which means you should save the file as license.txt in the ADF installation directory.

Non-trial users will receive a host-locked license file from SCM. To generate this file, SCM needs some information from you about your machines.

Windows users just need to double-click on the makelicinfo.bat script file in the ADF installation folder and send us the licinfo.txt file generated by the script.

Mac users can generate this information by double clicking the ADF2013.01 application. It will automatically compose an email with ther required information if no valid license is found. Alternatively, if the environment has been set up like a Unix-like system, the Unix procedure may be used.

Unix-like users can generate this information by running the following command at the shell prompt in a terminal window:

```
$ADFBIN/dirac info
```

Output of this command from all computers on which ADF will be running, including all nodes of a cluster if applicable, must be sent to license@scm.com.

In the case of very large clusters, it is often sufficient if you send us the output of the *$ADFBIN/dirac info* command from the head node and 2 or 3 compute nodes. As most compute node names have the same beginning, we can then add them with a wild card (for example Linuxchem*). It is important when you use this facility, to let us know that the info you are sending are not all compute nodes. Otherwise the license will only run on these few compute nodes.

After receiving this information SCM will prepare a license file matching your license conditions and e-mail it to you with instructions on how to install it.

After receiving your license file you will need to save it as described in the first paragraph of this section.

In a multi-user environment, make sure that permissions on the license file allow read access for everyone who is going to run ADF.

The following section contains important instructions on how to set up floating license. The instructions are simple but it is important to follow them exactly.

2.4.1 Floating license

If you have a floating license, you will need to follow the instructions below to set it up. Otherwise you may skip this section. Please note that floating licenses are not supported on Windows.

Create a floating license directory

Make a new directory, for example FloatADF, to keep track of the running ADF processes.

This FloatADF directory must be:

- in a fixed location (the directory name should not change!)
- shared between / mounted on all nodes where you want to run ADF
- writable by all users that want to run ADF
- FloatADF must **not** be a subdirectory of $ADFHOME

For example:

```
cd /usr/share
mkdir FloatADF
chmod 1777 FloatADF
```

If you also have a floating license for other modules, you need to set up different directories and repeat this procedure for all these modules (for example FloatBAND, FloatReaxFF, FloatDFTB, ...)

In the example, we have given all users read, write and execute permissions to this directory. If you wish to restrict this for security reasons, you may do so as long as all ADF users will have read, write and search permission for this directory. You may create an ADF user group for this purpose.

Important: the directory should **not** be a sub-directory of $ADFHOME as the directory name will change if you update to a new version! Also, do **not** put the license.txt file in the FloatADF directory.

The FloatADF directory may not be moved, deleted or renamed for the duration of your license because these actions will invalidate it!

E-mail us the license information

Send the result of the following commands (using again the name FloatADF and the location /usr/share as an example) to license@scm.com:

```
cd /usr/share
ls -lid $PWD/FloatADF
```

Please note that output of the ls command must include the full path of the FloatADF directory.

Together with the output of the ls command above, you also need to send us output of the *$ADFBIN/dirac info* command from each computer on which ADF will possibly run, as the license file is still host-locked.

In the case of very large clusters, it is often sufficient if you send us the output of the *$ADFBIN/dirac info* command from the head node and 2 or 3 compute nodes. As most compute node names have the same beginning, we can then add them with a wild card (for example Linuxchem*). It is important when you use this facility, to let us know that the info you are sending are not all compute nodes. Otherwise the license will only run on these few compute nodes.

2.5 Set up the scratch space

Most programs from the ADF package use disk for temporary data. This data often takes a significant amount of space and is used frequently. To avoid run-time errors and performance loss you may want to make sure that the file system used for temporary files is both big and fast. The SCM_TMPDIR environment variable is used to tell the programs where to put their temporary data. Please note that SCM_TMPDIR should always be set. If it is not set then each process will create its own directory in the current working directory where is was started.

Please see Section 2.3 and Appendix A on additional information about the SCM_TMPDIR variable.

Using multiple disks

Sometimes, if you have multiple non-RAID disks in your system, you may want to spread scratch files across different physical disks for better performance. It is possible to request that every ADF MPI-rank creates its files in a different directory by adding "%d" in $SCM_TMPDIR. If a "%d" string is encountered in the value of SCM_TMPDIR variable it will be replaced by the MPI rank number of the process at run-time. This means, however, that you may have to create up to 128 or more (depending on the maximum number of processes in one parallel calculation) symbolic links on each node where ADF is supposed to run. You should also create a directory matching the SCM_TMPDIR's value literally so that any process that does not interpret '%d' could also run.

Example: suppose there are two scratch file systems, /scratch1 and /scratch2 located on two different physical disks of an 8-core workstation. We want the odd rank numbers to use /scratch2 and the even numbers to use /scratch1. One way to achieve this is to create an empty /scratch directory and create nine symlinks in it as follows:

```
ln -s /scratch1 /scratch/%d
ln -s /scratch1 /scratch/0
ln -s /scratch2 /scratch/1
ln -s /scratch1 /scratch/2
ln -s /scratch2 /scratch/3
ln -s /scratch1 /scratch/4
ln -s /scratch2 /scratch/5
ln -s /scratch1 /scratch/6
ln -s /scratch2 /scratch/7
```

After that set SCM_TMPDIR to "/scratch/%d" and the ranks 0, 2, 4, 6 will use /scratch1 while ranks 1, 3, 5, and 7 will use /scratch2. When running ADF on a cluster it is better to combine multiple disks in a RAID 0 (striping) configuration as you probably do not want to create hundreds of symbolic links on each node.

2.6 Test your installation

This is a very important step and it should never be skipped.

2.6.1 Check the license file

Windows users test their license by double-clicking the makelicinfo.bat file in the ADF installation folder.

Unix users: first check that the license file has been installed correctly by running (at the shell prompt):

```
$ADFBIN/dirac check
```

This should produce the output similar to the following:

```
Checked: /home/testadf/adf2013.01/license.txt

License termination date (mm/dd/yyyy):  7/10/2013

According to it, you are allowed to run:

   ADF version  2013.99
  BAND version  2013.99
ADFGUI version  2013.99
BANDGUI version  2013.99
   GUI version  2013.99
 Utils version  2013.99
   NBO version  2013.99
```

```
   CRS version  2013.99

Number of procs you are allowed to use for:

ADF  :     128 procs
BAND :     128 procs

=====================================================
release: 2013.01
:example.com:
:Linuxmycomputer00:11:22:33:44:55:
ncores :           4
=====================================================
LICENCE INFO READY
```

If the license check is successful (i.e. you get the "LICENCE INFO READY" message) you can move on. Otherwise check that the license file has been installed according to instructions in Section 2.5.

2.6.2 Run some basic tests

Verify that you can now run examples provided with ADF and that they give correct results. We recommend that you consult the Examples document for notes on such comparisons: non-negligible differences do not necessarily indicate an error. If you have a license for the GUI you can also run a few GUI tutorials as a test.

Note: the example *.run files are complete Bourne shell scripts that should be executed as such, they are **not** input files to be fed into any program. The easiest way to run them is using ADFjobs.

2.6.3 Test the GUI

If the GUI is included in your license, check that you can start any of the GUI modules.

UNIX users:

Enter the following command:

```
$ADFBIN/adfjobs
```

An ADFjobs window should appear.

Mac users:

Double click the ADF2013.01 application to start it. An ADFjobs window should appear.

Windows users:

Double click the ADFjobs icon to start it. An ADFjobs window should appear. If the window does not appear or appears after a long delay then check the ADF-GUI requirements and check the firewall rules that should allow local communication.

All users should now be able to start ADFinput via the SCM menu on the top left of the menu bar.

2.6.4 Test parallel execution

It is very important to make sure that computer resources are utilized with the maximum efficiency. Therefore, you should check that each ADF job uses all processors/cores allocated to it and that the network is not overloaded with the disk I/O traffic.

Typically, when you submit a parallel job to the batch system you have a possibility to specify how many processors per node (ppn) to use. If the batch system you are using allows this, then make sure that you request ppn equal to the number of physical cores on each node.
Note that recent AMD processors based on the so-called "Bulldozer" cores have one floating-point unit (module) per two physical cores. On these Bulldozer architectures, ADF will proabably perform best when you only run on half the physical cores. We suggest you test this by setting ppn accordingly.

It is also possible that your batch system does not allow you to specify ppn but instead it always assumes that there only one processor per node. In this case, you will need to edit the $ADFBIN/start file and add some commands for processing the hostfile.

In order to check that everything is working as intended, create a reasonably small ADF job and start it, preferrably on more than one node. After the job has finished, open the job's .out file and find the table that looks like the following:

```
 Parallel Execution: Process Information
 ================================================================================
 Rank   Node Name                                NodeID   MyNodeRank  NodeMaster
    0   compute-0-0                                 0           0           0
    1   compute-0-0                                 0           1          -1
    2   compute-0-0                                 0           2          -1
    3   compute-0-0                                 0           3          -1
    4   compute-0-1                                 1           0           1
    5   compute-0-1                                 1           1          -1
    6   compute-0-1                                 1           2          -1
    7   compute-0-1                                 1           3          -1
 ================================================================================
```

Check the names in the "Node Name" column and verify that the number of tasks per node is correct. If it is, then move on to the next section.

2.6.5 Test parallel performance

If the process allocation to nodes looks as expected then scroll down a bit to the following lines:

```
Communication costs MPI_COMM_WORLD:      1.026 usec per message,   0.0320 usec
per 8-byte item
Communication costs for intra-node:      1.023 usec per message,   0.0318 usec
per 8-byte item
```

Make sure that you get reasonable numbers (< 100 usec per message) both for intra-node and MPI_COMM_WORLD communication costs. Otherwise contact your system administrator. If only the MPI_COMM_WORLD value is large but the intra-node one looks reasonable, this may mean that the network link between machines is very slow and you may want to run single-node jobs only.

If all seems to be OK then move to the end of the file to the table called "Timing Statistics". Before the table, there is a line of text beginning with "Total Used" that might look as follows:

```
 Total Used :   CPU=     8064.87      System=     1144.31     Elapsed=
9212.99
```

This shows how much time (in seconds) was spent in the ADF code (CPU) and in the kernel (System), and how long did it take for the calculation to complete (Elapsed). Ideally, the system time should be small

compared to the CPU time and the latter should be close to the Elapsed time. The system time will not always be a small number, however, the sum of the System and CPU times should always give a number very close to the Elapsed time. If this is not the case then it means that ADF has to spend a lot of time waiting for something. This can be, for example, disk I/O or network communication.

If you notice that the system time portion is enormously large or that there is a large difference between the CPU+System and the Elapsed time, then repeat the job with the "PRINT TimingDetail" keyword in the input and contact the SCM support.

2.7 Configure ADFjobs queues and 3rd party software (optional)

If you would like to use the GUI as a front-end to submit your jobs to queues, you should configure those queues in ADFjobs. Third party software MOPAC and ffmpeg (export movies) may also be installed separately.

2.7.1 ADFjobs queues

ADFjobs can submit and monitor jobs for you on different queues, both locally and remotely. You can use the GUI on your desktop or laptop for creating, submitting and analyzing your jobs, while running the jobs on remote compute clusters. In that case you should establish and ssh-agent connection (2.7.2), to enable remote job managing by the GUI. If you just run local jobs without a queuing system, skip this section.

To set up a new batch queue, select **Queue** → **New...** → and choose an example queuing system to starting with (LSF, PBS, or SGE). Modify the input files such that they correspond to your queue configuration:

- change the **Name** of the queue
- set the **Remote host** to the hostname of your cluster
- set **Remote user** to your username on that cluster
- change the **Run command** in accordance with your typical queue set up. The **$options** corresponds to the input box that can be modified in ADFjobs before submitting, e.g. the time or number of nodes
- you can set what $options defaults to in the **Default Options** field
- change the **Kill**, **Job status**, **System status commands**, if necessary
- you may define a **Prolog** or **Epilog** command

2.7.2 Managing remote jobs

To manage remote jobs, you need automatic authentication with an ssh-agent via an ssh key pair. This is most easily set up on MacOS or Linux, but is a bit more involved for Windows.

2.7.2.1 ssh connection on MacOS and Linux

If you don't have an ssh key pair already, you can generate one in a terminal: ssh-keygen -t rsa
Put your pass-word protected public key on the compute cluster(s) in ~/.ssh/authorized_keys.

Next, you should establish a connection via an ssh-agent. On MacOS an ssh-agent runs by default, and with keychain you just have to type your password once when you make the first connection.

On various Linux installations ssh-agent is also running by default. If an ssh-agent daemon is not yet running in the terminal where you will run ADFjobs, start the daemon, e.g. by adding eval $(ssh-agent bash) to your .bashrc. You need to add your private key to the ssh-agent daemon: ssh-add.

2.7.2.2 ssh connection on Windows

PuTTY tools, automatically installed with ADF, can be used to set up an ssh-agent connection. Follow these steps, using the PuTTY programs located in the bin\Putty directory in your ADF installation directory.

- Run puttygen.exe and follow the instructions to generate an ssh key pair. Make sure you use a password. Put the public key on the remote host in ~/.ssh/authorized_keys
- Run pageant.exe and add your private ssh key (right-click on the Pageant icon in the task bar) with your password
- Open a Command Prompt and go the Putty directory (e.g. cd C:\ADF2013.01\bin\Putty). Run plink user@host (with correct username and hostname) and type **y** when prompted to store the key in cache
- In the same Command Prompt, check that you can now connect through the ssh-agent:
  plink -batch -agent -l user host uptime

Now you can close the prompt and start using ADFjobs to manage remote jobs. You may test your queue configuration: **Jobs** → **Generate Test Job** and assign it to your new queue (2.7.1) before running the test job.

2.7.3 Centrally defined queues

A system administrator may also define a queue centrally, which may then be picked up by any user automatically via Dynamic queues. Consult the GUI manual for information on how to set these up in ADFjobs.

2.7.4 MOPAC

MOPAC is included with the ADF distribution. However, it needs to be enabled in your license file. If it is not enabled, please contact SCM for more information. Note that MOPAC is free for academic users but the MOPAC key has a limited duration. The MOPAC included with the ADF distribution is just the standard MOPAC from OpenMOPAC, and is updated frequently.

If you wish to use a MOPAC version different from the one included with the ADF distribution, you can do this by setting the SCM_MOPAC environment variable, either in your shell startup script or via the **SCM** → **Preferences** command:

- do not set SCM_MOPAC when you want to run the MOPAC included with the ADF pacakage, in most situations this is the easiest solution
- set SCM_MOPAC to the complete path to the Mopac executable
- set SCM_MOPAC to a command if you want to run MOPAC on a different machine, the command must pass the arguments and standard input, and should start the mopac.scm script on the other machine (located in $ADFBIN on that machine)

2.7.5 ffmpeg

The ADFmovie GUI module can make MPEG videos of the system as you see it in the ADFmovie window. This can be an MD trajectory or vibrational modes, or the course of geometry optimization. For this to work, you need to install the free ffmpeg program from the FFmpeg website and make sure it can be found in the path. The simplest way to do this is to copy the ffmpeg executable to $ADFBIN.

3. Compiling ADF from Sources

Compiling ADF from sources by end users is not supported on Windows. The following instructions apply to Linux/Unix and Mac OS X only.

3.1 Unpacking the distribution

Installing ADF with recompilation in mind is somewhat different from the binary-only installation. The downloaded source and binary tarballs must be unpacked in the following order (using Platform-MPI on x86_64-linux in this example):

```
# First sources
tar xzf adf2013.01.src.tgz
# Then binaries
tar xzf adf2013.01.pc64_linux.platform_mpi.bin.tgz
```

The result will be a adf2013.01 directory containing both binaries (for your platform) and sources.

3.2 Setting up environment

In addition to the standard environment variables discussed in the Section 2.3, you may need to set additional ones depending on your platform:

- MPIDIR: may be needed in the case of compiling ADF with non-default MPI, for example Intel-MPI. The MPIDIR variable is ignored when configuring with Platform-MPI.
- MATHDIR: on all x86 platforms this should be set to the the MKL root folder. If MKLROOT is defined and MATHDIR is not, then MKLROOT will be used.

3.3 Running Install/configure

After unpacking everything and setting up your environment properly, you need to run the configure script. This script is located in the $ADFHOME/Install directory, and it must be executed from the $ADFHOME directory. The script replaces some files in the bin directory with versions specifically targeted for your system. For example, if you want to install an OpenMPI version of ADF on a platform for which only a Platform-MPI version is available, then *Install/configure* will make an $ADFBIN/start script suitable for OpenMPI and overwrite the old one which was made for Platform-MPI. Further, *configure* creates the Makeflags and settings files that you will need to compile ADF.

Example:

```
cd $ADFHOME
Install/configure -p platform_mpi
```

The configure script accepts the following options:

- **-h** with this option configure will print the system name, processor type and, for Linux, the GLIBC version. It will also print the default machine name used internally by ADF and a list of known parallel implementations for this combination, as well as a list of possible optional parameters, if any (see below).

- **-p par** specifies the parallel implementation to use. The list of possible **par** values varies per platform and can be obtained by running configure with the **-h** option. The default is system-dependent and is the standard MPI for this processor type and operating system.
- **-o option** specifies an optional parameter if available. For example, on Power-PC/Linux you can specify *-o 32* to configure a version with 32-bit addressing.

3.4 Compiling ADF

Next, you need to compile the sources by executing yam (Yet Another Make) located in $ADFBIN

```
cd $ADFHOME
bin/yam
```

After a while, depending on the speed of your computer, everything should be ready to use, just as if you had installed the precompiled executables but now with your modifications included. Use *bin/yam -h* to get a list of yam command-line switches.

4. Additional Information and Known Issues

4.1 More on running MPI jobs

MPI (Message Passing Interface) is a standard describing how to pass messages between programs running on the same or different machines.

MPI is a formal standard and it is actively supported by all major vendors. Some vendors have highly-optimized MPI libraries available on their systems. There are also a couple of open-source implementations of the MPI standard, such as MPICH and OpenMPI. There are also numerous commercial MPI implementations that support a wide range of systems and interconnects, for example, Platform-MPI and IntelMPI.

Support for a particular MPI implementation in ADF can be considered at three levels: the source code, the configure script, and pre-compiled binaries. At each level different MPI implementations may be supported.

The ADF source code is not implementation-specific and thus theoretically it supports any MPI library. Many popular MPI implementations are supported at the level of the configure script. For example on 32-bit Linux these are: MPICH1, Intel MPI, OpenMPI, Platform-MPI. This means that proper compiler flags will be used and and an appropriate $ADFBIN/start script will be generated at configure time.

However, when choosing an MPI implementation for pre-compiled binaries, SCM considers many factors including (but not limited to) the re-distribution policy, performance, and built-in support for modern interconnects. Platform-MPI is currently the standard MPI implementation supported by SCM because it has the most favorable combination of these factors at this moment. For platforms where Platform-MPI is supported it is distributed with ADF. A different MPI implementation will be standard on a platform where Platform-MPI is not available. It may or may not be distributed with ADF. For example, SGI MPT is standard on SGI machines and OpenMPI is standard on Mac OS X platforms, but only the latter is distributed together with ADF.

IntelMPI has been added to the list of MPI implementations supported by SCM in 2013.

When pre-compiled binaries do not work on your computer(s) due to incompatibility of the standard MPI library with your soft- and/or hardware, the SCM staff will be glad to assist you in compiling ADF with the MPI implementation supported on your machine(s).

If you are going to use an MPI version of the ADF package, and it is not Platform-MPI or OpenMPI, you will need to determine if the corresponding MPI run-time environment is already installed on your machine. If not, you will need to install it separately from ADF. As it has been already mentioned, Platform-MPI and OpenMPI are bundled with the corresponding version of ADF so you don't need to worry about installing them separately.

Running with MPI on more than one node

When running on more than one machine (for example on a cluster **without** a batch system) you need to specify a list of hosts on which mpirun needs to spawn processes. In principle, this is implementation-specific and may be not required if the MPI is tightly integrated with your operating and/or batch system. For MPICH1 and Platform-MPI, you can do this by preparing a file containing hostnames of the nodes (one per line) you will use in your parallel job. Then you set the SCM_MACHINEFILE environment variable pointing to the file.

When you submit a parallel job to a batch system the job scheduler usually provides a list of nodes allocated to the job. The $ADFBIN/start shell script has some logic to extract this information from the batch system and pass it to the MPI's launcher command (typically mpirun). In some cases, depending on your cluster configuration, this logic may fail. If this happens, you should examine the $ADFBIN/start file and edit the

relevant portion of it. For example, you may need to add commands that process the batch system-provided nodelist or change mpirun's command-line options or even replace the mpirun command altogether.

4.2 More about Platform MPI

The use of Platform MPI is governed by our standard End User License Agreement (EULA)

As it has been already mentioned, Platform MPI is currently the standard MPI implementation on some systems, namely under Windows and on Intel-based Linux machines.

For more information about Platform MPI, including User's Guide please visit the MPI page at Platform Computing. Platform MPI is distributed with the Platform-MPI versions of ADF and a complete Platform-MPI directory tree is found in $ADFBIN/platform_mpi. Using Platform-MPI with ADF does not require any additional fees. In addition to TCP/IP and shared memory supported by every MPI implementation, Platform-MPI also supports, without recompilation, all well-known interconnects such as Infiniband and Myrinet. The best available interconnect is chosen automatically at run time and this can also be modified using mpirun command-line switches in the $ADFBIN/start script.

A few words about the mpirun commands found in the start script. If you look inside the $ADFBIN/start file on Linux, you will likely see three mpirun commands. They differ in the way the list of nodes is specified, if specified at all. The mpirun command with a *-lsb_hosts* switch is used under the LSF batch system. In employs the Platform-MPI's tight integration with LSF and lets it pick up the job configuration directly from the batch system.

The mpirun command with a *-hostfile* switch is used for multi-node interactive jobs and for all parallel jobs started under PBS or SGE batch systems. In the case of a batch system, the hostfile provided by the scheduler is passed to the mpirun command as-is, that is without modifications. The latter can be a problem if the format of the hostfile is different from that supported by Platform-MPI, in which case it should be modified accordingly. Please see the Platform-MPI User's Guide for information about supported hostfile formats.

Finally, the simplest form of the mpirun command is used for single-node jobs when the calculation is performed on the localhost. In this case the NSCM environment variable determines how many parallel processes will be started. Please note that if "$NSCM" is not equal to "1" (digit one) then the exact value of the NSCM variable has no effect when the job is started under any of the batch systems or with a hostfile. In this case the number of parallel processes to start is determined by the contents of the hostfile.

Remote shell command under Platform-MPI

One important point about Platform-MPI is that it needs a remote shell command, such as ssh, to start tasks on compute nodes. By default ssh is used but this can be changed using the MPI_REMSH environment variable. For example, executing the following from the bash prompt will make mpirun use rsh instead of ssh:

```
export MPI_REMSH=rsh
```

As usual, if you want to make the change permanent, you need to add this command to a shell resource file.

There are cases when a Linux cluster is configured in such a way that both ssh and rsh communication from/to/between compute nodes is disabled. One of the most common examples is TORQUE with the MAUI scheduler. In this case, there is a remote shell replacement utility called *pbsdsh*. This utility checks that it is executed under PBS and, if yes, allows you to start remote programs only on the nodes allocated to that particular job. In principle, this is all that mpirun needs. The only problem is that *pbsdsh* uses different command-line options. To solve this, we provide in $ADFBIN a script called *torque_ssh*. To make Platform-MPI always use torque_ssh instead of ssh simply set MPI_REMSH in the shell resource file as follows (after ADFBIN has been defined):

```
export MPI_REMSH=$ADFBIN/torque_ssh
```

Known Platform-MPI issue: libibverbs error message

Sometimes Platform-MPI stops with the following (or similar) error message

```
libibverbs: Fatal: couldn't open sysfs class 'infiniband_verbs'.
```

This occurs, for example, on GigE-connected ROCKS clusters with OpenMPI installed. The error is caused by the fact that there are Infiniband libraries installed without corresponding kernel drivers and/or hardware. In this case, one has to enforce the use of TCP by Platform-MPI, which can be done using one of the methods below:

- edit the $ADFBIN/start script and add a -TCP option to all mpirun commands:

  ```
  $ADFBIN/platform_mpi/bin/mpirun.mpich -TCP ...
  ```

- or set the MPIRUN_OPTIONS environment variable in your shell resource file (for example, ~/.bash_profile):

  ```
  export MPIRUN_OPTIONS=-TCP
  ```

- or set the MPI_IC_ORDER environment variable in your shell resource file (for example, ~/.bash_profile):

  ```
  export MPI_IC_ORDER="TCP"
  ```

Any of these options will make sure all other interconnects but TCP are ignored.

4.3 IntelMPI and SLURM

To get IntelMPI work under SLURM one needs to edit the $ADFBIN/start script and change the value of the I_MPI_PMI_LIBRARY environment variable to point to a correct libpmi library from SLURM.

4.4 Corrupted License File

You may find that, after having installed the license file, the program still does not run and prints a message "LICENSE CORRUPT". There are a few possible causes. To explain how this error may come about, and how you overcome it, a few words on license files.

Each license file consists of pairs of lines. The first of each pair is text that states in a human-readable format a couple of typical aspects: A 'feature' that you are allowed to use (for instance 'ADF'), the expiration date, a (maximum) release (version) number of the software and so on. The second line contains the same information in encrypted format: a long string of characters that appear to make little sense. The program reads the license file and checks, with its internal encrypting formulas, that the two lines match. If not, it stops and prints a "LICENSE CORRUPT" message.

So, there are two common reasons why this may happen:

1. You are using a license file for a version of the software other than your executables correspond to. Newer (major) releases may use a different encryption, so that the match in old license files is not recognized anymore. In particular, the "LICENSE CORRUPT" error will arise if you run ADF1999 (or later) with a license file that was generated for ADF2.3 (or earlier). Verify that your license file and executable belong to the same major release.

2. The license file as it has been created has been modified in some way. Sometimes, people inspect it and 'clean it up' a little bit, for instance by removing 'redundant spaces', or by making some other 'improvements'. Any such modification will break the encryption match and lead to the "LICENSE CORRUPT" error. Sometimes the reason lies in the mailing system: if the file contains long lines the mailer may break them into shorter lines. To verify (and correct) this open the license file in a text editor and see if it really consists of pairs of lines as described above. If not, re-unify the broken lines and try again.

You can use the **fixlic** utility to try to fix this automatically. Please be aware that the **fixlic** utility will try to fix the file pointed to by the $SCMLICENSE environment variable and replace it with the fixed copy. Thus, you need to make a backup of your license file first and you need to have write permissions for it.

```
cp $SCMLICENSE $SCMLICENSE.backup
$ADFBIN/fixlic
```

4.5 Windows: running jobs from the command line

In order to run ADF or any other program from the package without the GUI, navigate to the ADF installation directory and double click the adf_command_file.bat file. It will start a Windows command interpreter and set up the environment specific for that installation of ADF. Once it has started, cd to your jobs directory by entering the following commands at the prompt:

```
C:
cd \ADF_DATA
```

Then, run your job as follows (assuming the job is called h2o):

```
sh h2o.job
```

You can also prepare a job from a .adf file and run it using only two commands:

```
sh adfprep -t h2o.adf -j h2o > h2o.job
sh h2o.job
```

Please note that you do need to use *sh* in the commands above because both h2o.job and adfprep are shell scripts and, thus, they must be interpreted by a shell.

Appendix A. Environment Variables

The following environment varables must always be set for all ADF versions.

ADFHOME: full path of the ADF installation directory, for example, $HOME/adf2013.01.

ADFBIN: full path ADF directory with binaries, typically set to $ADFHOME/bin.

ADFRESOURCES: full path of the directory with the ADF database (basis sets and so on), typically set to $ADFHOME/atomicdata

SCMLICENSE: full path-name of the license file, for example $ADFHOME/license.txt.

SCM_TMPDIR: full path of the directory where all processes will create their temporary files. See also Section 2.5 and the special section on SCM_TMPDIR below.

The following environment variable may be required at run-time by a parallel version.

NSCM: The number of processes to be used in a particular calculation. This variable should only be set per job. Do **not** add any NSCM definitions to your shell resource files. Please note that the NSCM value is typically ignored in job submitted through a batch system because then the number of processors is determined by the batch job's properties.

SCM_MACHINEFILE full path-name of the file containing a list nodes to use for a job; **Important:** this variable should **only** be set if multiple computers are used without any batch system and then it should be set on the per-job basis. The file pointed to by the variable must already exist and it must contain a list of nodes on which a particular job is about to be executed, possibly with their processor count.

The following environment variables are relevant for the GUI modules.

SCM_ERROR_MAIL: e-mail address for error reports

SCM_GUIRC: location of the preferences file, by default $HOME/.scm_guirc

SCM_GUIPREFSDIR: location of the preferences folder, by default $HOME/.scm_gui/ (available since ADF2013.01b)

SCM_TPLDIR: location of the templates directory, by default no extra templates are loaded

SCM_STRUCTURES: location of the structures directory, by default no extra structures are loaded

SCM_RESULTDIR: location of the results directory, by default the current directory used

DISPLAY: specifies the X-window display to use and is required for all X11 programs on Linux/Unix and Mac OS X. On MaC OS X you should typically not set it as it will be set automatically. Setting it will break this.

SCM_MOPAC: command to start MOPAC, by default the $ADFBIN/mopac.scm shell script will be used

SCM_QUEUES: path to the dynamic queues directory, by default ADFjobs will search the remote $HOME/.scmgui file

The following environment variables are relevant for source distributions only, and only at the configure time.

MPIDIR and MATHDIR: see Section 3.2

The following environment variables may be set to modify other aspects of ADF execution. All of them are optional and some are used for debugging only.

SCM_IOBUFFERSIZE

Most programs within the ADF package use the KF I/O library. This library has a built-in caching mechanism that keeps parts of the files in memory. This allows to reduce the amount of the disk I/O significantly. The default buffer size depends on the platform and is typically set to 64 megabytes, which should be sufficient for running small jobs without much disk I/O. In some cases you can have a major performance improvement by making this buffer much larger, for example 512MB. You can do this by setting the SCM_IOBUFFERSIZE environment variable to a number corresponding to the buffer size **in megabytes**.

Please try for yourself, with your typical calculation on your production machine to find out the optimal value for your situation.

SCM_VECTORLENGTH

Almost all programs within the ADF package use numerical integration, and this is normally the most time-consuming part of the code. Numerical integration involves summing together results for each 'integration point'. The best performance is achieved when handling a number of points at the same time. The number of integration points handled together is called the block length or the vector length.

If the block length is too small, you will have a significant overhead and the programs may become very slow.

If the block length is too large, lots of data will not fit in the processor cache and again the program will not run at the maximum speed.

The optimal block length is somewhere in between, ranging from 32 to 4096 depending on your hardware. Sometimes it pays off to set the block length explicitly NOT to a power of 2 to avoid memory bank conflicts.

Again, try it yourself with your typical calculation on your production machine to find out the optimal value for your situation. On most machines, the default 128 is a good value.

SCM_SHAR_EXCEPTIONS: setting this variable to "*" disables the use of shared arrays.

SCM_DEBUG: setting this to a non-empty string will cause each MPI rank to print values of relevant environment variables and some messages about copying files to/from SCM_TMPDIR.

SCM_NOMEMCHECK: setting this to a non-empty string disables checks on memory allocation failures. The usefulness of this variable is questionable.

SCM_NODOMAINCHECK: setting this to a non-empty string disables DNS requests when verifying the license. Use this variable if you experience long delays at the start of each calculation.

SCM_TRACETIMER: setting this to a non-empty string will produce additional output on entry/exit to/from internal timers.

SCM_DEBUG_ALL: setting this to yes is equivalent to specifying DEBUG $ALL in the input

More on the SCM_TMPDIR variable

Below we will explain in more detail how does the SCM_TMPDIR environment work.

Every parallel job consists of one master and one or more slave tasks. Master and slaves behave a bit differently with respect to their scratch directories.

Slave processes

Slave processes will always create a directory for their scratch files in $SCM_TMPDIR and *chdir* to it to avoid any risk that shared files are updated by more that one process at the same time. For efficiency reasons, that directory should reside on a local disk unless you are using very, very fast shared file system for large files. You need write access to that directory, and the file system should have enough free space. Please note that the SCM_TMPDIR environment variable will be passed from the master to slaves.

After the job is finished, slave processes will delete their scratch directories. This can disabled by setting the SCM_DEBUG environment variable to any text, for example, to "yes". In this case the scratch directory and all its contects will be left intact. This directory will also be left behind when a job has crashed or has been killed.

Each slave writes its text output to a file called KidOutput located in its scratch directory. In case of an error this file will likely contain some sensible error message. If an error occurs and a slave process exits in a cotrollable way then in order to avoid losing the file ADF will copy the file to the directory, from which the job was started, as KidOutput__#, where # is the process' rank.

Master process or serial runs

The master process (which is the only process in a serial run) will also create its temporary files in its own sub-directory of $SCM_TMPDIR. There are some exceptions. Some files, such as logfile and TAPE13, will be created in the directory where ADF was started because they are not performance-critical but are convenient to have in the current directory for different reasons. For example, logfile is nice to have in the current directory in order to follow the calculation progress and the TAPE13 is an emergency restart file that can be used if ADF crashes or is killed.

At the end of a calculation, the master will copy all result files from its scratch directory to the directory where it was started.

Appendix B. Directory Structure of the ADF Package

Below is the list of major parts of the ADF package.

The bin directory

When the package is installed, the executable files are placed in $ADFHOME/bin. The binary executable file is usually called 'adf.exe', 'band.exe', and so on. On Windows it is adf.3.exe, band.3.exe, etc. There will also be files called just 'adf' or 'band'. The latter are shell scripts that execute the corresponding binaries.

You should use the script files, rather than the executables directly. The script files take care of several convenient features (like the BASIS key described in the ADF User's Guide) and provide a correct environment for running in parallel. See also the sample run scripts and the Examples document.

The $ADFBIN/start script takes care of setting up the environment and starting the binaries. If necessary, it parses the information provided by a batch system and sets up a machinefile (or an appfile) and runs tasks in parallel. Edit the file if you need to customize the way MPI programs are started.

The atomicdata directory

The directory atomicdata/ contains a large amount of data needed by programs from the ADF package at run time. For example, it contains basis sets for all atoms. Generally you should **not** modify any of the files in this directory.

The basis sets are described in detail in the ADF manual and BAND manual

The examples directory

The directory examples/ contains sample script and output files. The files with extension .run are shell scripts that also contain embedded input. Thus, these files should be executed, and not given as input to ADF.

The example calculations are documented in the ADF Examples documentation, and BAND Examples documention.

The source code (adf, band, libtc, Install, ...)

The source code files found in the program and library directories and subdirectories thereof have extensions .d, .d90, .dmod* or .cd and need to be pre-processed by the parser (scu) before compilation. The parser produces files with a suffix .f, .f90 or .c containing plain FORTAN or C code. These files are placed in the FFILES directory in each program's or library main directory at the compile-time. Other source files (used by the parser) are include files (files with extension .fh or .h) and files that define configuration parameters (the file "settings").

Parsing and compilation are controlled by *$ADFBIN/yam*.

The Install directory contains a configure script, some data files which provide generic input for configure (start, starttcl, and some more), a portable preprocessor cpp (based on Mouse cpp), the adf preprocessor (adfparser) and files that will be addressed by the *$ADFBIN/yam* command.

Apart from a few more files that are of little importance in the installation, you'll find in Install/ a number of subdirectories with names like pentium_linux_ifc, etc.: they contain precompiled binaries for supported platforms.

The Doc directory

All the user documentation for ADF is present in the PDF format in $ADFBIN/Doc. The same documentation is also available on the SCM website

The scripting directory

This directory contains some useful scripts that are part of the ADF package but are distributed under the GNU General Public License.