IDC/FLEX_EXTRACT/SIP
13 June 2018

English only

FLEX_EXTRACT V7.0.3 Software Installation Plan

This document defines the FLEX_EXTRACT V7.0.3 installation plan.

Summary

Flex_extract V7.0.3 is a collection of python (2.7) shell scripts and FORTRAN programs to retrieve input files for the FLEXTRA/FLEXPART Atmospheric Transport Modelling system from the ECMWF MARS archive.

The scripts can be 1) executed locally using the ECMWF WebMARS interface, or 2) on the ECMWF member state gateway server or HPC facility using the ecaccess software package. As another option the scripts can 3) also be triggered using ecflow scheduling software used by ECMWF for data dissemination.

This document describes the installation procedure and the software requirements necessary for installation. All required software used is open source and free of charge. Please note that the retrieval scripts have been completely rewritten using python as script language instead of ksh. The mode of usage has changed as well compared to earlier versions to reflect the additional functionality. Thus reading this document is recommended also to experienced flex_extract users.

The SIP also describes where log files can be found in case the scripts submitted via ecaccess or via ECMWF batch facilities fail.

Document history

Version	Date	Author	Description
0.1	13 July 2003	Alexander Beck	Initial draft
2.0	25 January 2007	Leopold Haimberger	Revision to describe installation procedure for ECMWFDATA V2.0
3.0	03 October 2008	Leopold Haimberger	Revision to describe installation procedure for ECMWFDATA V3.0
4.0	30 November 2010	Leopold Haimberger	Revision to describe installation procedure for ECMWFDATA V4.0
5.0	28 June 2012	Leopold Haimberger	Revision to describe installation procedure for ECMWFDATA V5.0
6.0	20 December 2013	Leopold Haimberger Anne Philipp	Revision to describe installation procedure for ECMWFDATA V6.0
7.0	1 December 2015	Leopold Haimberger Anne Philipp	Revision to describe installation procedure for ECMWFDATA V7.0
7.0.3	13 June 2018	Leopold Haimberger Anne Philipp	Revision to describe installation procedure for flex_extract_v7.0.3

Contents

1. SCOPE

1.1. Identification

This document applies to the flex_extract version 7.0.3.

1.2. System overview

Flex_extract is software that supplies the International Data Center Atmospheric Transport Modelling (IDC ATM) system with input data describing the state of the atmosphere and thus the current transport conditions. The system retrieves the data from the European Centre for Medium-Range Weather Forecasts (ECMWF) situated in Reading, United Kingdom. The system complements the NCEPDATA software already applied at IDC as another source for input data for ATM modelling.

1.3. Document overview

This installation plan describes the set of tasks necessary to install and test the flex_extract V7.0.3 software so it can be used operationally at IDC.

This document lists the sites where the software is to be installed. It defines the pre-requisites for installation in terms of hardware, software and security. It also describes the roles required to perform the installation including the skills and experience needed for each role. Finally, it describes the installation process including backup, installation, acceptance testing, rollback procedures and reporting.

The installation process is described as a sequence of numbered steps. Each step contains a description of what needs to be done. Unless otherwise stated, the success criterion for each step is that no error messages are generated.

After the step description, there is usually a sequence of commands that can be entered to complete the step. These commands are displayed using the `Courier font`. **These commands are just examples and may need to be adapted for different environments**.

This document is mainly intended for the installer, the deployment manager and the project manager. This document is also of interest to senior management and user representatives.

This document is compliant with the IDC Software Documentation Framework (2002) and the CTBTO Editorial Manual (2002).

2. REQUIREMENTS AND CONSTRAINTS

2.1. Installation environment

2.1.1. Installation sites

As outlined below the retrieval scripts will be in most cases installed at both the IDC and the ECMWF Linux member state server (ecgate). Installation on the ECMWF HPC is also possible.

2.1.2. Hardware environment

Flex_extract can be installed on both UNIX and LINUX platforms. Access to the Internet and no blocking firewalls are required.

2.1.3. Software environment

Flex_extract is a collection of python scripts and modules. It needs a standard python 2.6 or 2.7 environment on the local machine with numpy and dateutils as optional packages. Note that python must be compiled with '-fPIC' to be able to load the grib_api modules.

Flex_extract software can be used under three scenarios:

1) "REMOTE": Directly on the ECMWF Linux member state server ecgate (via ecaccess.ecmwf.int).
    a. Under this scenario only access to ecgate is needed.
2) "GATEWAY": Scripts are installed on a local machine and are submitted to ECMWF computer facilities via a member-state gateway server. Under this scenario the following software is needed:
    a. a UNIX/LINUX environment with an implementation of the Korn-shell (ksh).
    b. Web access tools provided by ECMWF (called ECaccess) : https://software.ecmwf.int/wiki/display/ECAC/ECaccess+Home
        i. Web Toolkit for different platforms:

https://software.ecmwf.int/wiki/display/ECAC/Packages

ii. Local gateway server have to be installed:

https://software.ecmwf.int/wiki/display/ECAC/Releases+-+Gateway+package

(The ECMWF gateway server cannot provide all required functionalities.)

3) "LOCAL": Scripts are installed and executed on a local machine. Under this scenario a software environment similar to that at ECMWF is required. In addition to the above requirements, the following software must be available:
    a. a working FORTRAN90 compatible compiler. The installation has been tested with the freely available GNU Fortran compiler (gfortran) version 4.4 or higher, as it is available on the ECMWF member state server ecgate or with the Intel compiler (ifort).
    b. ECMWF EMOS and GRIB API libraries, which can be downloaded from (https://software.ecmwf.int/wiki/display/EMOS/Emoslib) and (https://software.ecmwf.int/wiki/display/GRIB/Home).
    c. A python-based MARS client, which can be also downloaded https://software.ecmwf.int/wiki/display/WEBAPI/Access+MARS

While the "GATEWAY" scenario was by far the most common one in previous versions of flex_extract, it is likely that users may mostly use option "LOCAL" since it avoids usage of ecgate and no gateway must be installed. Option "LOCAL" is however limited to retrievals with size <20GB and causes more traffic over the Internet.

2.2. Installation constraints

A connection to the Internet must exist. If the "GATEWAY" scenario is used, the firewall at the site must be configured accordingly (port 9000 to ecaccess.ecmwf.int must be open). If not please ask your system administrator.

2.3. Security and privacy

In order to install the software, a user account is necessary both at IDC as well as at the ECMWF computer system. For scenarios “REMOTE” and “GATEWAY” a regular ECMWF user account must exist. For scenario “LOCAL” a web user id at ECMWF, which can be gained by a simple registration procedure, is sufficient.

2.4. Risk management

The software creates a directory flex_extract_v7.0.3 on ecgate unless otherwise specified. No files in this directory are removed but files from a previous installation of flex_extract_v7.0.3 are overwritten without further notice. Care has been taken not to overwrite any files from flex_extract version 1.0-6.0. The software requires 20MB of disk space, so there is little risk of exceeding the disk quota on the ECMWF member state server ecgate.

3. INSTALLATION PROCESS

3.1. Pre-installation procedures

To ensure a smooth installation for scenarios "REMOTE" and "GATEWAY", access to ecgate and availability of ecaccess tools should be verified. If there is no valid certificate, users should create it using the `ecaccess-certificate-create` command. Users should know their ECMWF user and group IDs and, if they plan to use ectrans, should also know their gateway server address and a valid ectrans destination. If there are problems please see section 2.1.3 to install/update the necessary software.

To ensure a smooth installation for retrieval via the ECMWF Web API, the user should verify that the GRIB API module can be loaded:

```
import gribapi
```

The ECMWF Web API can be tested by loading the ecmwfapi module:

```
import ecmwfapi
```

If this fails the user must install the python module as well as register and download the Web API key. If there are problems see section 2.1.3 and please follow the instructions provided at *https://software.ecmwf.int/wiki/display/WEBAPI/Access+MARS* to accomplish that.

3.2. Installation procedures

The flex_extract V7.0.3 software is distributed as one tar-file `flex_extract_v7.0.3.tar` containing python scripts, FORTRAN90 source code files, as well as (template) configuration files and a test suite. Ideally flex_extract V7.0.3 is installed as part of an existing FLEXPART installation. The `flex_extract_v7.0.3` directory should be in the directory FLEXPART_ROOT/scripts (see Fig. 1 for an illustration of the directory tree).
In this case the default directory settings in flex_extract will be optimal. Otherwise some may have to be set as command line parameters or in control files.

On ecgate or on the HPC the software will be installed in the $HOME directory.

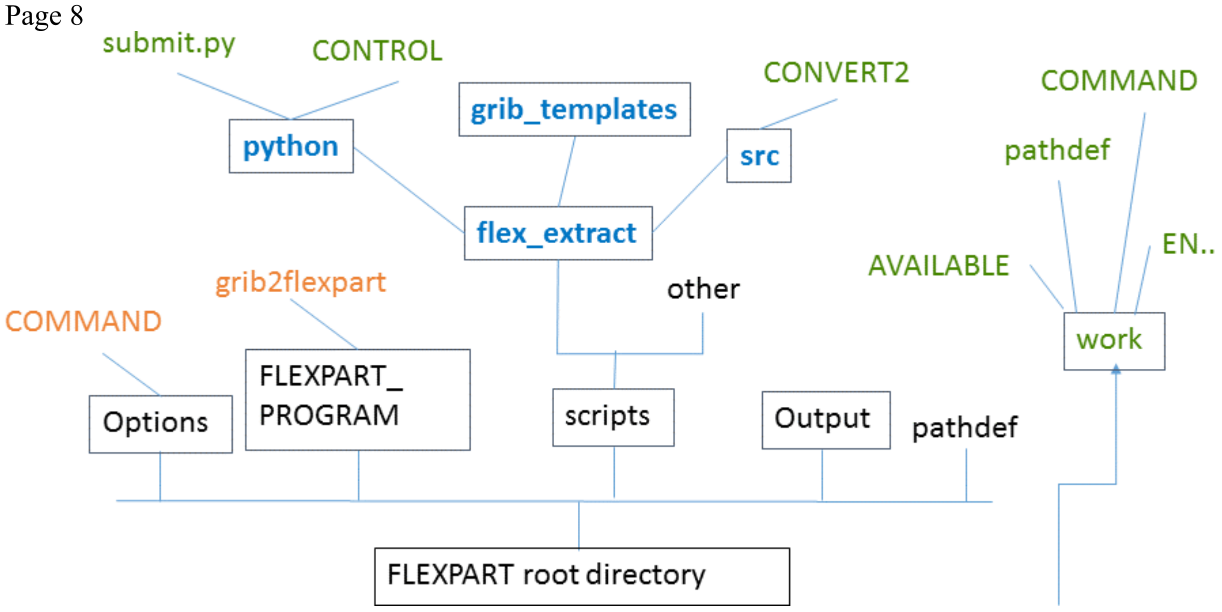

Fig. 1: FLEXPART directory structure (black) with directories in rectangles, files as text. flex_extract software should reside in scripts directory. In order to create input files on height levels (.fp files) flex_extract will need to have access to COMMAND file and grib2flexpart executable (red). Flex_extract software will create directory work and place all output files there. Usually the work directory will reside on a separate large file system.

The tar file contains the directory tree flex_extract_v7.0.3 as shown in Fig. 1. So the tar file should be extracted in the FLEXPART_ROOT/scripts directory. The flex_extract_v7.0.3 directory contains the directories and files:

- `python`, which contains the following scripts and CONTROL files
  - `install.py` is the python installation script for both local and remote software installation. Its set of options will be discussed below.
  - `submit.py` is the main python script for retrieving data via job submission or via the ECMWF WebAPI. Its set of options will be discussed below.
  - `testsuite.py` can be used to make some test retrievals.
  - `plot_retrieved.py` can be used to make simple test plots of the retrieved data
  - `Other *.py` contain functions and modules used by submit.py. `getMARSdata.py` may be called as standalone script to just retrieve the MARS data, `prepareFLEXPART.py` may be called as standalone script to do the flux deaccumulation and etadot calculations, the conversion to GRIB2 and to .fp format if necessary. If called at ecgate it also copies the resulting files to the gateway server if requested.
  - `job.temp.o, job.temp, compilejob.temp` are template files read by submit.py for job submission to ECMWF. `job.temp` is for a normal retrieval job, `compile_job.temp` is a job for installation of flex_extract on ecgate or the HPC facility.
  - `CONTROL*` are files read by the above python scripts to control their execution. They have the same syntax as in the earlier flex_extract 6.0 version. Some parameters in the control files can be overridden with command line parameters (see below).

  - o `job.ksh` is only a temporary file that is overwritten every time a job is submitted to ECMWF. Its header is compatible with both the SLURM and PBS batch queue systems used on ecgate and the HPC facility, respectively.
- `src` contains the FORTRAN source files for CONVERT2, the executable that calculates the hybrid coordinate vertical velocity. There are several makefiles targeted for different installation scenarios. These may have to be adapted. The makefiles can be chosen via a command line parameter of install.py (see below).
- `grib_templates` contains a conversion table from alphanumeric MARS parameters to purely numeric paramIDs used in GRIB, as well as a description of the ERA-Interim Gaussian grid.

The recommended installation procedure depends on the availability of a gateway server. If one exists, the software can be installed without the need to log in interactively on ecgate.

3.2.1. Installation on the local machine

Even if it is intended to run the retrieval scripts on ecgate it is recommended to install them on the local machine. This is particularly true if FLEXPART is run on the local machine as well. Installation is simply to extract flex_extract_v7.0.3.tar in the $FLEXPART_ROOT/scripts directory as depicted in Fig. 1:

After the download just change directory to the installation directory and execute:
```
tar xvf flex_extract_v7.0.3.tar.gz
```

If the FORTRAN compiler, python 2.7, the ECMWF grib_api, the emoslib, as well as WebMARS are correctly installed, there is only one step left to do:

```
cd flex_extract_v7.0.3/python
setenv FLEXPART_ROOT path_to_FLEXPART
install.py  --target  local|ecgate|cca  [--flexpart_root_scripts
=$FLEXPART_ROOT/scripts] [--makefile Makefile.xxx]
```

This step compiles `CONVERT2` and checks if all needed software is installed. If `$FLEXPART_ROOT` is not provided, the `flex_extract_v7.0.3` directory tree is installed under the current directory in scenario ”LOCAL”, otherwise it is installed under `$HOME`.

For convenience it is recommended to add the `$FLEXPART_ROOT/scripts/flex_extract_v7.0.3/python` directory to your `$PATH` and `$PYTHONPATH` environment variables.

If compilation with the default makefile (Makefile.local.ifort for “LOCAL”, Makefile.gfortran for “GATEWAY”) fails, the user may try other provided makefiles. The settings in the makefiles should be identical to those used for building the GRIB API software to avoid version incompatibilities. If in doubt please ask your system administrator.

3.2.2. Installation using a gateway server

If a local gateway server exists or if there is no firewall between the ECMWF standard gateway server ecaccess.ecmwf.int and the local server, one can install the flex_extract V7.0.3

software on ecgate or cca without the need to login interactively on these machines. Of course the user must have the necessary credentials to login on ecgate or cca, however.

On the IDC system (suitable directory) execute:
```
tar xvf flex_extract_v7.0.3.tar
```

If necessary renew the ECACCESS certificate:
```
ecaccess-certificate-create
```
Enter your ECMWF user ID and `passcode`

Then start the installation with the following commands:
```
cd flex_extract_v7.0.3/python
install.py   --target   ecgate   --ecuid=ecuid   --ecgid=ecgid
--gateway=gateway         --destination=destination         [--
flexpart_root_scripts = ]
```
If the user has enough credentials, one may also specify one of the HPC machines (`cca,ccb`) as target.

The install script checks if ecaccess software is installed and if a gateway exists. Then it compiles `CONVERT2` on ecgate (using gfortran) and on cca (using the Intel ifort compiling environment) and, if `$FLEXPART_ROOT_SCRIPTS` is not provided, the `flex_extract_v7.0.3` directory tree is installed under `$HOME` on ecgate or cca.

When called as above, `install.py` creates a job (`compilejob.ksh`) from a template (`compilejob.temp`) and sends it to the respective ECMWF machine. The user should receive an email a few minutes after `install.py` has been called. One can monitor progress of the job locally via the command `ecaccess-job-list.`

The install script also creates a job template (`job.temp`, which contains the `ecuid,ecgid,gateway,destination` settings) from an original job template (`job.temp.o`). This job template is later used by the python script submit.py to create the actual jobs submitted to ECMWF, which are named `job.ksh`. The `destination` must be a valid target that has been defined by the user. See http://www.ecmwf.int/services/ecaccess/guide/Ectrans_setup.html for details on how to define a valid destination. The two variables are needed for the `ectrans` facility.

After this step the user should be able to submit on-demand or operational retrieval jobs via ecgate or cca.

Functionality of job submission via ecaccess can be tested by entering
```
submit.py --queue=ecgate --start_date=YYYYMMHH
submit.py --queue=cca --start_date=YYYYMMHH
```

Again one can monitor the progress of the scripts locally via the command
```
ecaccess-job-list
```
or on ecgate with the command
```
squeue | grep $USER
```
The user should also get an email notification after the script has been completed.

Submit.py also creates jobs ready to be called by the operational dissemination suite:

```
submit.py --queue=ecgate --start_date=YYYYMMHH --basetime= 00|
12
```

The temporary `job.ksh` files created by `submit.py` can be used as templates to be triggered by the operational suites of ECMWFs ECFLOW (former SMS) system.
The values YYYYMMDDHH and 00 or 12 are then provided by ECFLOW.

The example scripts send emails to `$ECUID`, notifying success and failure of the scripts. The example scripts leave data and log files in $SCRATCH/ms_sms_output_V7 (operational) or $SCRATCH/python??????? (on demand).
In addition, if the parameter ECSTORAGE is set to 1 in the control files, the script protocols are archived together with the output files in the ecfs file directories `ectmp:/$ECUID/econdemand` or `ectmp:/$ECUID/ecops`. For an introduction on ecfs see http://www.ecmwf.int/services/computing/training/material_2012/ecfs.pdf.
On the HPC facility the example scripts are executed in a temporary directory that exists only during job lifetime. Information about script success can be found in the email sent to the user or in the `/scratch/ms/$ECGID/$ECUID/.ecaccess_DO_NOT_REMOVE` directory.

3.2.3. Interactive installation on ecgate without a gateway server

Interactive installation on ecgate or cca can be accomplished the same way as on the local IDC server: In some download directory (e.g. `$SCRATCH`) execute

```
tar xvf flex_extract_v7.0.3.tar
```

The necessary software is surely installed on ecgate or cca but it must be activated using the module command:

```
  module load python
  module load grib_api
  module load emos
```

Then the installation is started as:

```
cd flex_extract_v7.0.3/python
install.py --target=local --makefile=Makefile.gfortran
```

Note that contrary to the submission via a member state server, the variables `GATEWAY` and `DESTINATION` need to be set as environment variables in ecgate or must be set in the `CONTROL` files.

3.2.4. Interactive installation on cca (the HPC facility)

Interactive installation on cca can be accomplished the same way as on the local IDC server: In some download directory (e.g. /home/ms/$GROUP/$USER) execute

```
tar xvf flex_extract_v7.0.3.tar
```

The necessary software is surely installed on ecgate or cca but it must be activated using the module command:

```
module switch PrgEnv-cray PrgEnv-intel
module load python
module load grib_api
module load emos
```

Note that the scripts expect the Intel programming environment. Then the installation is started as:

```
cd flex_extract_v7.0.3/python
install.py --target=local --makefile=Makefile.CRAY
```

The PBS settings in job.temp.o may be changed as appropriate. Execution on cca is generally recommended only for very demanding retrievals that fail or take too long on ecgate, e.g. when calculating the hybrid coordinate vertical velocity on the Gaussian grid using native (T1279 or higher) spectral resolution. For such cases, the HPC provides more memory (>10GB) than ecgate and allows multithreading (OMP_NUM_THREADS>1, e.g. 48).

3.3. Installation testing

3.3.1. Success criteria for scripts on local machine

After installation, one can retrieve data using the submit.py script, e.g.

```
cd $FLEXPART_ROOT_SCRIPTS/flex_extract_v7.0.3/python
submit.py --start_date=20131107
```

The user can follow the progress of the script. It should complete with no errors. Submit.py can be called from anywhere if its location is set in the $PATH variable. Note that also the Control file needed by the script must exist. If there is no control file in the current directory submit.py will look in `$FLEXPART_ROOT_SCRIPTS/flex_extract_v7.0.3/python.` If it is not found there it will fail. Also please note that there must be enough space for the working directory. At least 20GB are recommended. The default location for the working directory is `$FLEXPART_ROOT_SCRIPTS/flex_extract_v7.0.3/work` but can be changed with the command line option `--inputdir.`

For a more thorough testing the user may try the test suite:

```
cd $FLEXPART_ROOT_SCRIPTS/flex_extract_v7.0.3/python
testsuite.py [test_group]
```

It calls submit.py with common options. The optional command line parameter refers to different test cases which are defined in the file `testsuite.json.` Please note that the test suite is experimental. The test suite may take several hours to complete. Output can be found in the `flex_extract_v7.0.3/test` directory.

3.3.2. Success criteria for scripts via ecgate

After installation, the directory tree `$FLEXPART_ROOT_SCRIPTS/flex_extract_v7.0.3` with the python scripts and with the other files mentioned above must exist on the local IDC machine and at least also on ecgate.

The installation was successful if the example scripts (for ecgate) produce `ENyymmddhh` files and an email reports about the success.

The `ENyymmddhh` files are stored by default on `ectmp:/$USER/econdemand` or `ectmp:/$USER/ecops` and if `$ECTRANS` is set to 1, they are also sent to the IDC or other customers. For an introduction on ecfs and ectrans see e.g. http://www.ecmwf.int/services/computing/training/material_2012/ecfs.pdf and http://www.ecmwf.int/services/ecaccess/guide/Ectrans_setup.html.

The execution of the scripts after submission can be followed by issuing

```
squeue | grep $USER
```

on ecgate or

```
qstat -u $USER
```

on cca or by

```
ecaccess-job-list
```

on the local machine. Note that the scripts involve MARS requests which can take from a few minutes up to a few hours, depending on the load of the MARS system.

If the on demand scripts fail, one should look at the two log files produced by the scripts:

- `$SCRATCH/flex??????.out` and
- `$SCRATCH/python??????/prot`

Please note that only the latter file is also sent via email. The ?????? are job IDs or process numbers which are used to ensure that on demand scripts running at the same time do not overwrite each other. Normally the output of these two log files is sufficient to trace back the reason for the failure. If not one can go into the directory `$SCRATCH/python?????/` Once there one can try to repeat the job steps that failed. This may give further hints.

The output for the operational scripts can be found in files

- `$SCRATCH/ms_sms_output_V7/CTBTO_ops.out` and
- `$SCRATCH/ms_sms_output_V7/ecmwf_idc_ops_ecgate_V7_*_??`

Again only the latter file is also sent via email. Normally the output of these two log files is sufficient to trace back the reason for the failure. If not one can go into the directories named `$SCRATCH/ctbto_run_ecgate-?hrs-?` and there one can try to repeat the job steps that failed. If unsure which directory to use, try `ls -lrt` to find out which one was last modified. Please note that operational scripts overwrite the output of a previously run script. **Therefore only one operational script should run at a time!**

3.3.3. Installation checklist

The following checks should be performed

- Is there a directory `$FLEXPART_ROOT_SCRIPTS/flex_extract_v7.0.3` on ecgate and does it contain scripts, source code and the compiled executable `$FLEXPART_ROOT_SCRIPTS/flex_extract_v7.0.3/src/CONVERT2`? Note that in most cases `$FLEXPART_ROOT_SCRIPTS` will be equal to `$HOME`. If no, repeat steps 3.2.1 or 3.2.2.
- Submit some of the example scripts. After submission:
    - Are the scripts running? (see 3.3.1 how to check this)
    - Is there a new log file in `$SCRATCH` or `$SCRATCH/ms_sms_output_V7`?
- After the scripts have been completed:

- Did you get a notification via email? Did it report errors?.
  - If not, check the email and the job log files for error messages.
- Are there new files `ENyymmddhh` at the places noted in 3.3.1?
  - If not check the `ecuid,ecgid,gateway,destination` settings used when calling `install.py`
  - Note that the file suffix does not need to be EN. This is only the default. Depending on the configuration file parameter PREFIX it could be also EG or ES or something else.
- If ECTRANS is set to 1: Are there new files `ENyymmddhh` on the local gateway in the target directory specified in the ectrans setup?
  - If not check the paths set in the `CONTROL_OPS` or `CONTROL_ERA` configuration files

- After the scripts have been completed and tested:
  - The scripts are set such that they do not remove the working directory `$SCRATCH/python?????` on ecgate. Only if you specify the option `'--debug=False'` the working directory is removed. This is useful for debugging purposes but with time those residual files may fill up too much disk space on `$SCRATCH` so that the user has to remove directories from past retrievals.

For a more thorough testing the user may try the test suite:

```
cd $FLEXPART_ROOT_SCRIPTS/flex_extract_v7.0.3/python
testsuite.py
```

It calls submit.py with common options for retrieval via ecgate. Success of the test suite may be judged from the emails received and from the EN files transferred to the local server. The test suite may take several hours to complete.

3.4. Rollback procedures

The software installation procedure should not in any case cause failures of the overall system. The software can be removed completely by executing the command

```
\rm -r $FLEXPART_ROOT_SCRIPTS/flex_extract_v7.0.3
```

on *ecgate*.

4. DETAILS ON INSTALLATION SCRIPTS AND FILE TEMPLATES

This section describes how the installation scripts assemble the example scripts from several code snippets. The idea of this procedure is to duplicate only those parts of the scripts which are really volatile and should be changeable by the users while only one copy of the main part of the retrieval scripts exists. This leads to far more easily maintainable code. In the following, the update script is described, followed by a documentation of the script parts.

4.1. The installation script install.py

The script install.py is used to make flex_extract V7.0.3 ready for use both on the local machine and the ECMWF gateway and HPC.

```
[lh0@ecgb11 ~/flex_extract_v7.0.3/python]$ ./install.py -h
usage: install.py [-h] [--target INSTALL_TARGET] [--makefile MAKEFILE]
                  [--ecuid ECUID] [--ecgid ECGID] [--gateway GATEWAY]
                  [--destination DESTINATION]
                  [--flexpart_root_scripts FLEXPART_ROOT_SCRIPTS]
                  [--job_template JOB_TEMPLATE] [--controlfile CONTROLFILE]

Install flex_extract software locally or on ECMWF machines

optional arguments:
  -h, --help            show this help message and exit
  --target INSTALL_TARGET
                        Valid targets: local | ecgate | cca , the latter two
                        are at ECMWF (default: None)
  --makefile MAKEFILE   Name of Makefile to use for compiling CONVERT2
                        (default: None)
  --ecuid ECUID         user id at ECMWF (default: None)
  --ecgid ECGID         group id at ECMWF (default: None)
  --gateway GATEWAY     name of local gateway server (default: None)
  --destination DESTINATION
                        ecaccess destination, e.g. leo@genericSftp (default:
                        None)
  --flexpart_root_scripts FLEXPART_ROOT_SCRIPTS
                        FLEXPART root directory on ECMWF servers (to find
                        grib2flexpart and COMMAND file) Normally flex_extract
                        resides in the scripts directory of the FLEXPART
                        distribution, thus the: (default: None)
  --job_template JOB_TEMPLATE
                        job template file for submission to ECMWF (default:
                        job.temp.o)
  --controlfile CONTROLFILE
                        file with control parameters (default: CONTROL.temp)
```

The last three options are normally not necessary to be set. However please note that the install script relies on existence of job.temp.o and CONTROL.temp and compilescript.temp. These must not be removed and should not be changed.

4.2. The script template compilejob.temp

```
#!/bin/ksh

# ON ECGB:
# start with ecaccess-job-submit -queueName ecgb NAME_OF_THIS_FILE  on gateway server
# start with sbatch NAME_OF_THIS_FILE directly on machine

#SBATCH --workdir=/scratch/ms/spatlh00/lh0
#SBATCH --qos=normal
#SBATCH --job-name=flex_ecmwf
#SBATCH --output=flex_ecmwf.%j.out
#SBATCH --error=flex_ecmwf.%j.out
#SBATCH --mail-type=FAIL
#SBATCH --time=12:00:00

## CRAY specific batch requests
##PBS -N flex_ecmwf
##PBS -q ns
##PBS -S /usr/bin/ksh
# -o /scratch/ms/no/sbc/flex_ecmwf.$Jobname.$Job_ID.out
# job output is in .ecaccess_DO_NOT_REMOVE
##PBS -j oe
##PBS -V
##PBS -l EC_threads_per_task=1
##PBS -l EC_memory_per_task=3200MB

set -x
export VERSION=7.0
```

```
case $HOST in
  *ecg*)
  module load python
  module load grib_api
  module load emos
  export FLEXPART_ROOT_SCRIPTS=
  export MAKEFILE=Makefile.gfortran
  ;;
  *cca*)
  module switch PrgEnv-cray PrgEnv-intel
  module load grib_api
  module load emos
  module load python
  echo ${GROUP}
  echo ${HOME}
  echo $HOME | awk -F / '{print $1, $2, $3, $4}'
  export GROUP=`echo $HOME | awk -F / '{print $4}'`
  export SCRATCH=/scratch/ms/${GROUP}/${USER}
  export FLEXPART_ROOT_SCRIPTS=
  export MAKEFILE=Makefile.CRAY
  ;;
esac

mkdir -p $FLEXPART_ROOT_SCRIPTS/flex_extract_v$VERSION
cd $FLEXPART_ROOT_SCRIPTS/flex_extract_v$VERSION   # if FLEXPART_ROOT is not set this means cd
to the home directory
tar -xvf $SCRATCH/flex_extract_v$VERSION.tar
cd src
\rm *.o *.mod CONVERT2
make -f $MAKEFILE >flexcompile 2>flexcompile

ls -l CONVERT2 >>flexcompile
if [ $? -eq 0 ]; then
  echo 'SUCCESS!' >>flexcompile
else
  echo Environment: >>flexcompile
  env >> flexcompile
fi


mail -s flexcompile.$HOST.$$ $USER <flexcompile
```

Install.py adapts the job headers of this template and creates `compilejob.ksh`. This script is then submitted to ecgate or cca.

4.3. The script template job.temp.o

This file should never be removed. Install.py modifies the header of job.temp.o according to the parameters ecuid,ecgid and creates the job template job.temp. This template is then used by submit.py to create job.ksh which is actually submitted to ecgate or cca.

```
#!/bin/ksh

# ON ECGB:
# start with ecaccess-job-submit -queueName ecgb NAME_OF_THIS_FILE  on gateway server
# start with sbatch NAME_OF_THIS_FILE directly on machine

#SBATCH --workdir=/scratch/ms/spatlh00/lh0
#SBATCH --qos=normal
#SBATCH --job-name=flex_ecmwf
#SBATCH --output=flex_ecmwf.%j.out
#SBATCH --error=flex_ecmwf.%j.out
#SBATCH --mail-type=FAIL
#SBATCH --time=12:00:00

## CRAY specific batch requests
##PBS -N flex_ecmwf
##PBS -q np
##PBS -S /usr/bin/ksh
## -o /scratch/ms/spatlh00/lh0/flex_ecmwf.$PBS_JOBID.out
## job output is in .ecaccess_DO_NOT_REMOVE
```

```
##PBS -j oe
##PBS -V
##PBS -l EC_threads_per_task=24
##PBS -l EC_memory_per_task=32000MB

set -x

case $HOST in
  *ecg*)
  module load python
  module load grib_api
  module load emos
  export PATH=${PATH}:
  ;;
  *cca*)
  module switch PrgEnv-cray PrgEnv-intel
  module load grib_api
  module load emos
  module load python
  export SCRATCH=$TMPDIR
  export PATH=${PATH}:
  ;;
esac

cd $SCRATCH
mkdir -p python$$
cd python$$

export CONTROL=CONTROL

cat >>$CONTROL<<EOF
EOF

submit.py --controlfile=$CONTROL --inputdir=./work --outputdir=./work >prot

mail -s flex.${HOST}.$$ $USER <prot
```

4.4. The control file template CONTROL.temp

This is the default control file used by the retrieval script submit.py. It should never be removed.
Many but not all of the parameters in CONTROL.temp can be overruled by command line parameters of the script. The users can change it or create their own control files that are read if specified in the command line parameter --controlfile of submit.py. Note that the settings in CONTROL.temp will lead to a retrieval of ERA-Interim data on a rather coarse grid. This is convenient for testing purposes but likely not optimal for the user's needs. If the parameters in the control file are smartly set, one will have to specify only few command line parameters in submit.py. Please do not remove parameters in this file, however, since quite a lot of them are needed for the retrieval scripts to function properly.

```
DAY1
DAY2
DTIME 3
TYPE AN FC FC FC FC FC AN FC FC FC FC FC AN FC FC FC FC FC AN FC FC FC FC FC
TIME 00 00 00 00 00 00 06 00 00 00 00 00 12 12 12 12 12 12 18 12 12 12 12 12
STEP 00 01 02 03 04 05 00 07 08 09 10 11 00 01 02 03 04 05 00 07 08 09 10 11
CLASS EI
STREAM OPER
NUMBER OFF
EXPVER 1
GRID 5000
LEFT -175000
LOWER -90000
UPPER 90000
RIGHT 180000
LEVEL 60
LEVELIST 1/to/60
RESOL 63
GAUSS 1
```

```
ACCURACY 16
OMEGA 0
OMEGADIFF 0
ETA 0
ETADIFF 0
DPDETA 1
SMOOTH 0
FORMAT GRIB1
ADDPAR 27/28/173/186/187/188/235/139/39
PREFIX EN
ECSTORAGE 1
ECTRANS 0
ECFSDIR ectmp:/${USER}/econdemand/
MAILOPS ${USER}
MAILFAIL ${USER}
GRIB2FLEXPART 0
EOF
```

In the default setting, the example scripts do not use ectrans. Users have to manually set ECTRANS to one if they want to use that. This is a security measure in order to avoid jamming of transfers from multiple scripts in case ectrans settings are not correct.

4.5. The test suite setting in testsuite.json

This file in JSON format is read by the script testsuite.py to submit several test jobs. The tests are organised in groups (the red names). These names can be used as command line parameters in testsuite.py.

```
 {
"install": {
"script": "install.py",
"control": null,
"inputdir": null,
"ecuid":"lh0",
"ecgid":"spatlh00",
"gateway":"srvx7.img.univie.ac.at",
"destination":"leo@genericSftp",
"local": {"target":"local"},
"local_flexpart":
{"target":"local","flexpart_root_scripts":"~/TEST2_FLEX_EXTRACT_GRIB2_NF4/scripts"},
"ecgate": {"target":"ecgate"},
"cca": {"target":"cca"}
},
"default": {
"script": "submit.py",
"control": null,
"inputdir": null,
"start_date": "20131107",
"local": {},
"local_flexpart": {"flexpart_root_scripts":"~/TEST2_FLEX_EXTRACT_GRIB2_NF4/scripts"},
"ecgate": {"queue":"ecgate"},
"cca": {"queue":"cca"}
},
"work": {
"script": "submit.py",
"control": null,
"start_date": "20131107",
"inputdir": "$SCRATCH/work",
"local_flexpart": {"flexpart_root_scripts":"~/TEST2_FLEX_EXTRACT_GRIB2_NF4/scripts"}
},
"fc": {
"script": "submit.py",
"control": "CONTROL_FC",
"inputdir": "$SCRATCH/workfc",
"local_flexpart": {"flexpart_root_scripts":"~/TEST2_FLEX_EXTRACT_GRIB2_NF4/scripts"}
},
"cv": {
"script": "submit.py",
"control": "CONTROL_CV",
"inputdir": "$SCRATCH/workcv",
"start_date": "20131107",
"local_flexpart": {"flexpart_root_scripts":"~/TEST2_FLEX_EXTRACT_GRIB2_NF4/scripts"}
},
"hires": {
"script": "submit.py",
"control": "CONTROL_HIRES",
"inputdir": "$SCRATCH/workhires",
"local_flexpart": {"flexpart_root_scripts":"~/TEST2_FLEX_EXTRACT_GRIB2_NF4/scripts"},
"ecgate": {"queue":"ecgate"}
},
"hiresgauss": {
"script": "submit.py",
"control": "CONTROL_HIRESGAUSS",
"inputdir": "$SCRATCH/workhiresgauss",
"basetime": "00",
"local_flexpart": {"flexpart_root_scripts":"~/TEST2_FLEX_EXTRACT_GRIB2_NF4/scripts"},
"ecgate": {"queue":"ecgate"},
"cca": {"queue":"cca"}
},
"ops": {
"script": "submit.py",
"start_date": "20131108",
"control": "CONTROL_OPS_V6.0_4V.temp",
"inputdir": "$SCRATCH/workops2",
"local_flexpart": {"basetime":"00"},
"local_flexpart12": {"basetime":"12"},
"ecgate": {"queue":"ecgate","basetime":"00"}
},
"opsfc": {
"script": "submit.py",
"start_date": "20131108",
"control": "CONTROL_OPS_V6.0",
"inputdir": "$SCRATCH/workopsfc",
"local_flexpart": {"basetime":"00"},
"local_flexpart12": {"basetime":"12"},
"ecgate": {"queue":"ecgate","basetime":"00"}
}
}
```

Glossary

[This section defines terms that are specific to the project and are actually used in the current document.]

Abbreviations

[This section defines abbreviations that are actually used in the current document.]

CTBT	Comprehensive Test Ban Treaty
CTBTO	Comprehensive Nuclear-Test-Ban Treaty Organization
ECMWF	European Centre for Medium-Range Weather Forecasts
HPC	High Performance Computer
IDC	International Data Centre
MARS	Meteorological Archival and Retrieval System at ECMWF

REFERENCES

[List all documents that are referred to in the text e.g. "Schneider, G., Winters. J.P (2001). Applying Use Cases: A practical guide. Addison-Wesley." Documents should be listed in alphabetical author order.]

International Organisation for Standardization (ISO) (1995). Information technology – Software life cycle processes. ISO/IEC 12207.

Comprehensive Nuclear-Test-Ban Treaty Organization (CTBTO) (2002). Editorial Manual.

International Data Centre (IDC) (2002). IDC Software Documentation Framework.