



AWS Workshop: GNSS RO and Cloud Computing

Stephen Leroy¹, Amy McVey¹, Mark Leidner¹, Hailing Zhang², Eric Wang³

OPAC-7 / IROWG-9, Leibnitz, Austria

September 13, 2022

¹Verisk Atmospheric and Environmental Research, Lexington, Massachusetts, USA

²COSMIC, University Corporation for Atmospheric Research, Boulder, Colorado, USA

³Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, USA

Agenda

1. Motivation

- Cloud computing with AWS

2. Getting started

- Logging in to EC2 instance (AWS virtual machine)
- Jupyter notebooks

3. GNSS RO data in the AWS Open Data Registry

- Open Data Registry and Github support
- AWS command line interface

4. Data assimilation with JEDI-FV3

5. AWS Universe applied to GNSS RO data

Motivation

- Cloud computing with Amazon Web Services
- Cloud computing with Amazon Web Services using GNSS RO data
- Cloud computing with Amazon Web Services using GNSS RO data with Jupyter notebook and brief lectures
- Cloud computing with Amazon Web Services using GNSS RO data with Jupyter notebook and brief lectures without cost

Getting started with your virtual machine/EC2 instance

MacOS and Linux

1. Download the irowg_2022.pem file; `chmod 700 irowg_2022.pem`
2. `% ssh -i irowg_2022.pem -L 2222:localhost:22 -L 8888:localhost:8888 ec2-user@<...>amazonaws.com`
3. `[ec2-user] % jupyter notebook --no-browser --port=8888`
4. Copy last URL into your favorite browser

Sent in your personalized email!

Windows

1. Download the irowg_2022.ppk file
2. Install and launch an X server (e.g., Xming) and PuTTY
3. In PuTTY...
 - Set hostname = `ec2-user@<...>amazonaws.com` port = 22, connection type = SSH
 - Connection >> SSH >> X11, check "Enable X11 Forwarding"
 - Connection >> SSH >> Auth, select "Private key file for authentication" and browse for irowg_2022.ppk file
 - Connection >> SSH >> Tunnels: Source port = 8888, Destination = localhost:8888
 - Open the connection
4. `[ec2-user] % jupyter notebook --no-browser`
5. Copy last URL into your favorite browser

Jupyter notebooks

- Open the `jupyter_intro.ipynb` notebook and try a few python commands
- Open `workshop.ipynb` jupyter notebook
- Feel free to download any material directly to your laptop using “File >> Download”. We will post the `workshop.ipynb` on the website after this workshop.

What is Amazon Web Services (AWS)?

- Cloud potential: Limitless compute and storage
- Cloud caveat #1: Set your own limits, or it will cost you!
- Cloud caveat #2: Develop a test plan, increase scale methodically.
- Services: Packaged resources with a specific purpose (over 200)

The screenshot displays the 'All services' page on the AWS website. It features a header 'All services' and a sub-section 'Services by category'. The services are organized into six columns, each with an icon and a category title. The 'Compute' column lists EC2, Lightsail, Lambda, Batch, Elastic Beanstalk, Serverless Application Repository, AWS Outposts, EC2 Image Builder, and AWS App Runner. The 'Containers' column lists Elastic Container Registry, Elastic Container Service, Elastic Kubernetes Service, and Red Hat OpenShift Service on AWS. The 'Storage' column lists S3, EFS, FSx, and S3 Glacier. The 'Quantum Technologies' column lists Amazon Braket. The 'Management & Governance' column lists AWS Organizations, CloudWatch, AWS Auto Scaling, CloudFormation, Config, OpsWorks, Service Catalog, Systems Manager, AWS AppConfig, Trusted Advisor, Control Tower, AWS License Manager, AWS Well-Architected Tool, AWS Health Dashboard, AWS Chatbot, Launch Wizard, AWS Compute Optimizer, and Resource Groups & Tag Editor. The 'Security, Identity, & Compliance' column lists IAM, Resource Access Manager, Cognito, Secrets Manager, GuardDuty, Inspector, Amazon Macie, AWS IAM Identity Center (successor to AWS Single Sign-On), Certificate Manager, Key Management Service, CloudHSM, Directory Service, WAF & Shield, AWS Firewall Manager, Artifact, Security Hub, Detective, AWS Signer, AWS Network Firewall, and AWS Audit Manager.

AWS basic services you are using

- EC2 – Elastic Cloud Compute
 - Physically, it is a set of compute nodes on a rack in a data center in a region.
 - Groups of machines are broken out into “families” each providing a characteristic ratio of CPU cores to RAM
 - CPU and RAM Examples

Type	Family	vCPU	GB RAM	On-Demand \$/hr
c5.xlarge	C5	4	8	\$0.17
r5.24xlarge	R5	96	768	\$6.048
t3.medium	T3	2	4	\$0.0416

- S3 – Simple Storage Service (Unlimited Storage)
 - Can only store objects, so no symbolic links and no folders. Each object has a specific prefix.
 - Pricing tiers exist for cost savings potential
- Identity and Access Management (IAM)
 - IAM Roles allow users and services permission via IAM Policies to allow or deny actions or have condition-based permissions.

AWS command line interface

Local machine requirements: Python 3.8+ with awscli

“conda install -c anaconda awscli”, or “pip install awscli”

```
aws --no-sign-request s3 ls s3://gnss-ro-data/contributed/v1.1/ucar/
```

```
aws --no-sign-request s3 ls \  
    s3://gnss-ro-data/contributed/v1.1/ucar/metop/atmosphericRetrieval/2009/06/01/
```

```
aws --no-sign-request s3 cp --recursive \  
    s3://gnss-ro-data/contributed/v1.1/romsaf/metop/atmosphericRetrieval/2009/06/01/ ./
```

See [aws_commands.txt](#) provided in email or [aws_commands](#) in EC2 instance.

How to access the AWS Open Data Registry in Python

- **S3FS: Linux commands for AWS in Python**
 - This allows you to link an s3 bucket as a file system.
 - Enables easy use of ls and cp commands to the files stored on s3.
- **Boto3**
 - Allows you to create, search, and modify any AWS resource and its metadata within they python language.

NASA ACCESS 2019

- GNSS RO data in the AWS Open Data Registry
 - <https://registry.opendata.aws/gnss-ro-opendata/>
 - s3://gnss-ro-data, region: “us-east-1”, *unsigned authentication*
 - RO data in “contributed/”
 - Database data in “dynamo/”
- Support material in GitHub
 - <https://github.com/gnss-ro/aws-opendata/>
 - Documentation of RO data in repository, including detailed PDFs
 - Utilities to aid in manipulating the data, especially creating a private DynamoDB database for RO data
 - Tutorial demonstrations: Database queries, inter-center comparison, tropopause analysis

DynamoDB Service

- A serverless database, key-value NoSQL
- Must have a unique Partition Key and Sort Key combo
- Pay for what you use and Easy to query

<input type="checkbox"/>	leo-ttt	date-time	gps_seconds	latitude	local_time	longitude	mission	occid	receiver
<input type="checkbox"/>	sacc-G03	2006-03-09-15-51	825954679	24.17	15.855	78.864	sacc	sacc-G03-2...	sacc
<input type="checkbox"/>	sacc-G03	2006-03-09-16-36	825957321	-54.438	16.589	-118.203	sacc	sacc-G03-2...	sacc
<input type="checkbox"/>	sacc-G03	2006-03-09-21-36	825975435	-23.238	21.621	170.068	sacc	sacc-G03-2...	sacc
<input type="checkbox"/>	sacc-G03	2006-03-09-23-19	825981613	-42.904	23.337	168.402	sacc	sacc-G03-2...	sacc
<input type="checkbox"/>	sacc-G03	2006-03-10-05-05	826002331	37.549	5.092	-102.621	sacc	sacc-G03-2...	sacc
<input type="checkbox"/>	sacc-G03	2006-03-10-10-51	826023079	-35.137	10.855	-11.389	sacc	sacc-G03-2...	sacc
<input type="checkbox"/>	sacc-G03	2006-03-10-15-40	826040295	-48.139	15.637	-82.931	sacc	sacc-G03-2...	sacc
<input type="checkbox"/>	sacc-G03	2006-03-10-16-36	826043814	31.29	16.615	78.109	sacc	sacc-G03-2...	sacc
<input type="checkbox"/>	sacc-G03	2006-03-10-17-22	826046433	-51.961	17.343	-146.332	sacc	sacc-G03-2...	sacc

Create your own copy of the DynamoDB database

<http://github.com/gnss-ro/aws-opendata/tree/master/utilities>

Simple but slow...

```
% python3 import_gnss-ro_dynamoDB.py --dynamodb_table_name my_ro_database \  
--mission champ --date_str "2003"
```

Fast...

Follow instructions in DynamoDB_full_import_instructions.txt

Data Analysis with Python: DynamoDB structure

Partition key	Sort key	Information
<i>leo-ttt</i>	<i>yyyy-mm-dd-hh-mm</i>	
metopa-G01	2020-01-01-00-47	<i>Satellite information:</i> transmitter, receiver, mission
	2020-01-01-01-51	<i>Geolocation information:</i> longitude, latitude, time, local_time, setting
	2020-01-01-02-41	
	...	<i>Paths:</i> ucar_calibratedPhase, ucar_refractivityRetrieval, ucar_atmosphericRetrieval, romsaf_refractivityRetrieval, romsaf_atmosphericRetrieval
metopa-G02	2020-01-01-01-10	
	2020-01-01-02-02	
	2020-01-01-03-14	
	...	
metopb-G01	2020-01-01-00-17	
	2020-01-01-01-21	
	2020-01-01-02-11	
	...	
metopc-G32	2020-01-01-00-19	
	...	
	2020-01-01-21-31	
	2020-01-01-22-24	
	2020-01-01-23-06	

- Querying requires **one specific value** for a partition key (receiver-transmitter).
- Querying requires sort key specification, but it can be loose (“between” two values).
- Querying can optionally filter results of query by RO “information” (metadata); compound filters allowed.

Some things to explore...

To discover what methods are available to you, ...

```
>> dir( s3 )
```

```
>> dir( Key )
```

```
>> dir( Attr )
```

```
>> from Resources import valid_missions
```

Change the processing center...

```
>> processing_center = "romsaf"
```

Try rising occultations...

```
>> filters = filters & Attr("rising").eq( "False" )
```

Running NWP on AWS instance – an example of JEDI

- **The goal is to conduct NWP experiments on AWS using the RO open data.**
 - It is not difficult!
 - We use JEDI for an example: Joint Effort for Data assimilation Integration (JEDI), developed by JCSDA with partners. <https://github.com/JCSDA-internal/>
- **Config and launch AWS instance**
 - Select appropriate instance: cost vs. requirement
 - Use existing Amazon Machine Image(AMI)
 - etc.
- **Conduct NWP experiment**
 - Activate the modules from AMI
 - Compile/build you system
 - Fetch data from s3 and prepare data for your NWP system.
 - **Converter/reader from RO open data to JEDI IODA format**
 - Conduct NWP experiments as you usually do on your HPC clusters!
- **Make your work sharable – Create your own AMI**



EC2 instance



The screenshot shows the AWS Management Console interface for the US East (N. Virginia) region. The left sidebar contains navigation links for Services, Features, Limits, AMIs, and Elastic IPs. The main content area is titled 'Console Home' and includes a 'Recently visited' list with links to EC2, CodeBuild, S3, and CloudFormation. Below this is a 'Welcome to AWS' section with links for 'Getting started with AWS' and 'Training and certification'. The right-hand side of the console displays the 'EC2 Global view' with a 'Resources' table, a 'Launch instance' button, 'Service health' information, and 'Account attributes'.

Resources			
You are using the following Amazon EC2 resources in the US East (N. Virginia) Region:			
Instances (running)	19	Dedicated Hosts	0
Elastic IPs	12	Instances	78
Key pairs	59	Load balancers	1
Placement groups	5	Security groups	107
Snapshots	86	Volumes	150

Launch instance	
Launch instance	Migrate a server

Service health	
Status	This service is operating normally

Account attributes	
Supported platforms	VPC
Default VPC	vpc-489db231
Settings	EBS encryption
Zones	EC2 Serial Console
	Default credit specification
	Console experiments

EC2 instance

The screenshot shows the 'Launch an instance' page in the AWS Management Console. The breadcrumb navigation is 'EC2 > Instances > Launch an instance'. The main heading is 'Launch an instance' with an 'Info' link. Below this is a brief introduction: 'Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.' The 'Name and tags' section has a text input for 'Name' containing 'hz_nwp_ro_opendata' and a link for 'Add additional tags'. The 'Application and OS Images (Amazon Machine Image)' section includes a search bar with 'aws-opendata-reader-ucar' and tabs for 'Recents', 'My AMIs', and 'Quick Start'. Under 'My AMIs', there are radio buttons for 'Owned by me' (selected) and 'Shared with me'. A 'Browse more AMIs' link is also present. A dropdown menu shows a selected AMI: 'skylab-1.0.0-ubuntu20-bufr-11.7.1' with its ID 'ami-0b24d818d257651f6' and other details like '2022-08-30T20:29:47.000Z', 'Virtualization: hvm', 'ENA enabled: true', and 'Root device type: ebs'.

September 13, 2022

The screenshot shows the 'Choose an Amazon Machine Image (AMI)' page. The breadcrumb navigation is 'Services > Search for services, features, blogs, docs, and m [Option+S]'. The main heading is 'Choose an Amazon Machine Image (AMI)'. Below this is an explanation: 'An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.' The search bar contains 'aws-opendata-reader-ucar'. Below the search bar are four categories: 'Quickstart AMIs (0) Commonly used AMIs', 'My AMIs (0) Created by me', 'AWS Marketplace AMIs (2551) AWS & trusted third-party AMIs', and 'Community AMIs (1) Published by anyone'. The 'Refine results' section has a 'Clear all filters' button and filters for 'Owner' (with 'Owned by me' checked) and 'OS category' (with 'All Linux/Unix' and 'All Windows' unchecked). The 'Publish date range' filter is also visible. The search results show 'aws-opendata-reader-ucar (1 filtered, 0 unfiltered)'. The selected AMI is 'aws-opendata-reader-ucar' with ID 'ami-03ef6e6e1ecb63692'. A description states: 'This image was built based on the skylab-1.0.0-redhat-demo. All copyright belongs to the JCSDA/UCAR.' The 'Select' button is highlighted. The AMI details include: 'Platform: Other Linux Architecture: x86_64', 'Owner: 469205354006', 'Publish date: 2022-08-28', 'Root device type: ebs Virtualization: hvm', 'ENA enabled: Yes', and 'Tags: name: aws-opendata-reader-ucar'.

AWS Workshop: GNSS RO in the Cloud

18

Config instance

▼ Instance type [Info](#)

Instance type

t2.micro

Family: t2 1 vCPU 1 GiB Memory
On-Demand Linux pricing: 0.0116 USD per Hour
On-Demand Windows pricing: 0.0162 USD per Hour

Free tier eligible

[Compare instance types](#)

▼ Key pair (login) [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

hz_nwp_ro_opendata

[Create new key pair](#)

▼ Network settings [Info](#) Edit

Network [Info](#)

vpc-4a9db231 | cdash-server

Subnet [Info](#)

No preference (Default subnet in any availability zone)

Auto-assign public IP [Info](#)

Enable

Firewall (security groups) [Info](#)

A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

Create security group

Select existing security group

Common security groups [Info](#)

Select security groups

[Compare security group rules](#)

Global SSH sg-0f715b41f30e6d3aa X
VPC: vpc-4a9db231

Security groups that you add or remove here will be added to or removed from all your network interfaces.

EC2 > Instances > Launch an instance



Success

Successfully initiated launch of instance (i-08d7bc407b4168327)

[▶ Launch log](#)

Next Steps

Get notified of estimated charges

[Create billing alerts](#) to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier)

How to connect to your instance

Your instance is launching and it might be a few minutes until it is in the running state, when it will be ready for you to use

Click [View Instances](#) to monitor your instance's status. Once your instance is in the 'running' state, you can connect to it from the [Instances](#) screen. Find out [how to connect to your instance](#)

[View more resources to get you started](#)

[View all instances](#)

Connect to instance



Instances (1/1) Info

Search

hz X Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv4 DNS
hz_nwp_ro_op...	i-0ca2c2aa701ab6442	Running	t2.micro	2/2 checks passed	No alarms	us-east-1a	ec2-107-20-109...

Instance: i-0ca2c2aa701ab6442 (hz_nwp_ro_opendata)

Public IPv4 address 172.31.22.41	Instance state Running	Public IPv4 DNS ec2-107-20-109-189.compute-1.amazonaws.com
Hostnames IP name: ip-172-31-22-41.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-22-41.ec2.internal	Elastic IP addresses -
Answer private resource DNS name IPv4 (A) 107.20.109.189 [Public IP]	Instance type t2.micro	AWS Compute Optimizer finding Opt-in to AWS Compute Optimizer for recommendations
IAM Role -	VPC ID vpc-4a9db231 (cdash-server)	Learn more
▼ Instance details Info	Subnet ID subnet-ff51d0b5	Auto Scaling Group name -
Platform Other Linux (Inferred)	AMI ID ami-03ef6e6e1ecb63692	Monitoring disabled
Platform details Red Hat Enterprise Linux with High Availability	AMI name aws-opendata-reader-ucar	Termination protection Disabled
Stop protection Disabled	Launch time Wed Aug 31 2022 23:24:04 GMT-0600 (Mountain Daylight Time) (7 minutes)	AMI location 469205354006/aws-opendata-reader-ucar
Instance auto-recovery Default	Lifecycle normal	Stop-hibernate behavior disabled
AMI Launch index 0	Key pair name hz_nwp_ro_opendata	State transition reason -

EC2 > Instances > i-0ca2c2aa701ab6442 > Connect to instance

Connect to instance Info

Connect to your instance i-0ca2c2aa701ab6442 (hz_nwp_ro_opendata) using any of these options

EC2 Instance Connect | Session Manager | **SSH client** | EC2 serial console

Instance ID
i-0ca2c2aa701ab6442 (hz_nwp_ro_opendata)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is hz_nwp_ro_opendata.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
`chmod 400 hz_nwp_ro_opendata.pem`
4. Connect to your instance using its Public DNS:
`ec2-107-20-109-189.compute-1.amazonaws.com`

Example:
`ssh -i "hz_nwp_ro_opendata.pem" root@ec2-107-20-109-189.compute-1.amazonaws.com`

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Start your work

```
(venv) [ec2-user@ip-172-31-22-41 skylab-demo]$ module list
Currently Loaded Modulefiles:
 1) stack-gcc/11.2.1
 2) numactl/2.0.14
 3) pmix/4.1.2
 4) zlib/1.2.12
 5) openmpi/4.1.3
 6) stack-openmpi/4.1.3
 7) stack-python/3.9.7
 8) cmake/3.22.1
 9) curl/7.83.0
10) tar/1.30
11) gettext/0.21
12) libunistring/0.9.10
13) libidn2/2.3.0
14) pcre2/10.39
15) git/2.35.2
16) hdf5/1.12.1
17) parallel-netcdf/1.12.2
18) netcdf-c/4.8.1
19) nccmp/1.9.0.1
20) netcdf-fortran/4.5.4
21) parallelio/2.5.4
22) wget/1.19.5
23) base-env/1.0.0
24) boost/1.78.0
25) ecflow/5.8.3
26) bufr/11.7.0
27) ecbuild/3.6.5
28) openjpeg/2.3.1
29) eccodes/2.25.0
30) eigen/3.4.0
31) openblas/0.3.19
32) eckit/1.19.0
33) fckit/0.9.5
34) ecmwf-atlas/0.29.0
35) git-lfs/2.13.3
36) gsl-lite/0.37.0
37) krb5/1.19.3
38) libtirpc/1.2.6
39) hdf/4.2.15
40) jedi-cmake/1.3.0
41) netcdf-cxx4/4.3.1
42) json/3.10.5
43) json-schema-validator/2.1.0
44) py-attrs/21.4.0
45) py-pycparser/2.20
46) py-cffi/1.15.0
47) py-findlibs/0.0.2
48) py-setuptools/59.4.0
49) py-numpy/1.22.3
50) py-eccodes/1.3.2
51) py-mpi4py/3.1.2
 52) py-h5py/3.6.0
 53) py-cftime/1.0.3.4
 54) py-netcdf4/1.5.3
 55) py-bottleneck/1.3.2
 56) py-numexpr/2.7.3
 57) py-six/1.16.0
 58) py-python-dateutil/2.8.2
 59) py-pytz/2021.3
 60) py-pandas/1.4.0
 61) py-pybind11/2.8.1
 62) py-pycodestyle/2.8.0
 63) py-pyhdfe/0.10.4
 64) py-pyyaml/6.0
 65) py-gast/0.5.3
 66) py-beniget/0.4.1
 67) py-ply/3.11
 68) py-pythran/0.11.0
 69) py-scipy/1.8.0
 70) udunits/2.2.28
 71) jedi-base-env/1.0.0
 72) py-jmespath/0.10.0
 73) py-urllib3/1.26.6
 74) py-botocore/1.21.12
 75) py-s3transfer/0.5.0
 76) py-boto3/1.18.12
 77) json-c/0.15
 78) sqlite/3.38.5
 79) proj/8.1.0
 80) gdal/3.4.3
 81) geos/3.9.1
 82) libpng/1.6.37
 83) py-cycler/0.11.0
 84) py-fonttools/4.31.2
 85) py-kiwisolver/1.3.2
 86) py-pyparsing/3.0.6
 87) py-packaging/21.3
 88) py-pillow/9.0.0
 89) py-matplotlib/3.5.2
 90) py-certifi/2021.10.8
 91) py-cython/0.29.30
 92) py-pyproj/3.1.0
 93) py-pyshp/2.1.0
 94) py-shapely/1.8.0
 95) py-cartopy/0.20.2
 96) py-markupsafe/2.0.1
 97) py-jinja2/3.0.3
 98) py-ruamel-yaml/0.17.16
 99) py-ruamel-yaml-clib/0.2.4
100) jedi-ewok-env/1.0.0
101) fms/release-jcsda
102) jedi-fv3-env/1.0.0
103) antlr/2.7.7
104) gsl/2.7.1
105) nco/5.0.6
```

source ./activate.sh

python awsro2ioda.py -i ref* -o output.nc4 -d
2021080502



Create your own AMI

Instances (1/1) Info

Search

hz X Clear filters

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
-	ami-03ef6e6e1ecb63692	Available	aws-opendata-reader-ucar	469205354006/aws-opendata-reader-...	469205354006	Public

Amazon Machine Images (AMIs) (1/1) Info

Owned by me Search

aws-opendata X Clear filters

Name	AMI ID	AMI name	Source	Owner	Visibility
-	ami-03ef6e6e1ecb63692	aws-opendata-reader-ucar	469205354006/aws-opendata-reader-...	469205354006	Public

AMI ID: ami-03ef6e6e1ecb63692

Details Permissions Storage Tags

AMI ID	Image type	Platform details	Root device type
ami-03ef6e6e1ecb63692	machine	Red Hat Enterprise Linux with High Availability	EBS
AMI name	Owner account ID	Architecture	Usage operation
aws-opendata-reader-ucar	469205354006	x86_64	RunInstances:1010
Root device name	Status	Source	Virtualization type
/dev/sda1	Available	469205354006/aws-opendata-reader-ucar	hvm
Boot mode	State reason	Creation date	Kernel ID
-	-	Sun Aug 28 2022 00:52:49 GMT-0600 (Mountain Daylight Time)	-
Block devices	Description	Product codes	RAM disk ID
/dev/sda1-snap-032e6f0ca27177e78:995:true:gp2	This image was built based on the skylab-1.0.0-redhat-demo. All copyright belongs to the JCSDA/UCAR.	-	-

