ADDE Server Concepts

Dave Santek and Russ Dengel

Space Science and Engineering Center

2016 McIDAS Users' Group Meeting
Madison,WI
14 November 2016

Goals

- Overview of ADDE concepts
  - What are ADDE servers?
  - Client/server system structure
  - Required components of ADDE servers
  - Required functionality of ADDE servers
  - What about calibration and navigation?

Topics

- Overview of Abstract Data Distribution Environment (ADDE)
  - Image data
  - Client and server aspects
  - Directory and data server
- McIDAS Area file structure

Topics

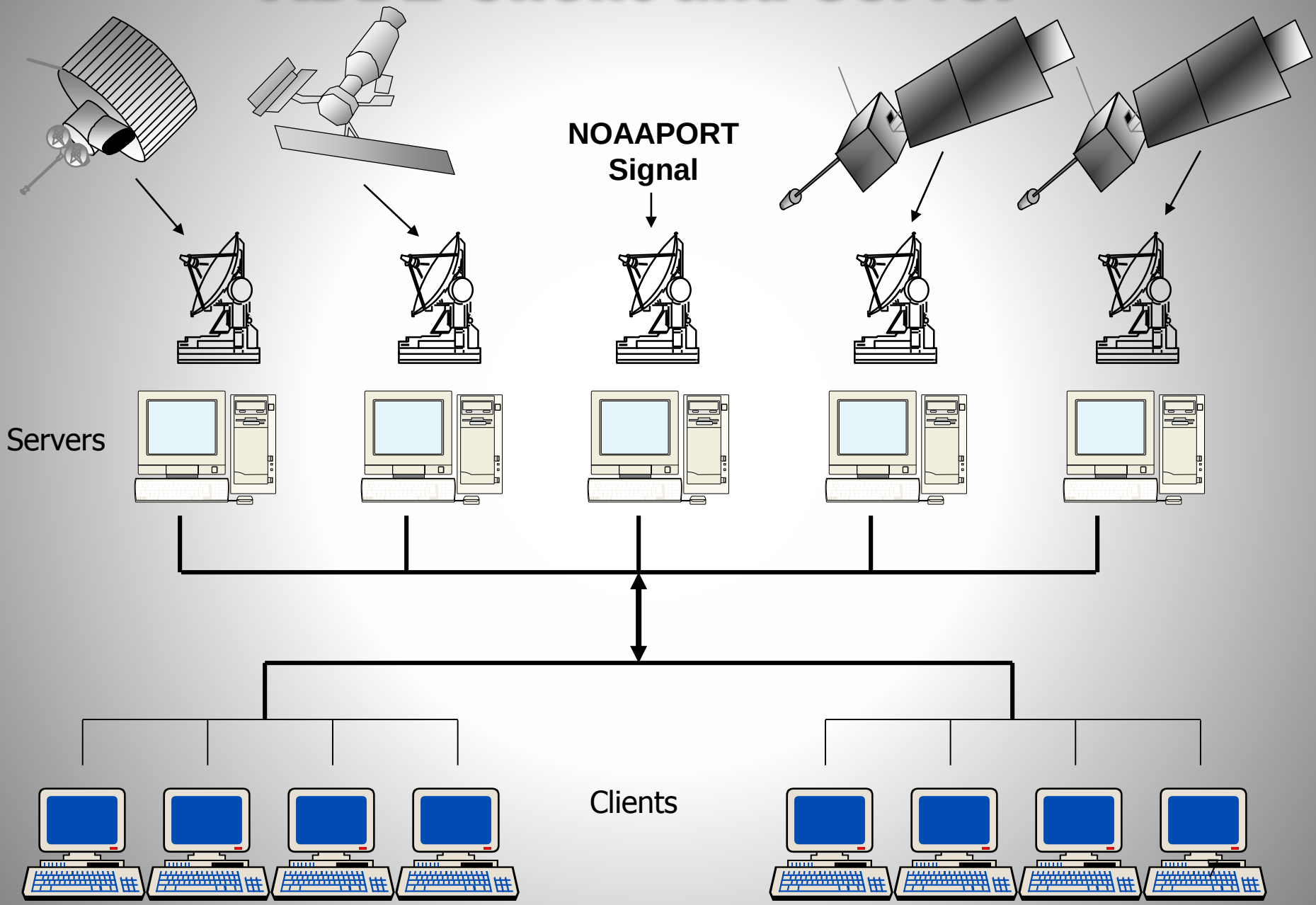
- ADDE low-level details
- Required components for ADDE servers
- Required functionality for ADDE servers
- Calibration and navigation modules

What are ADDE servers?

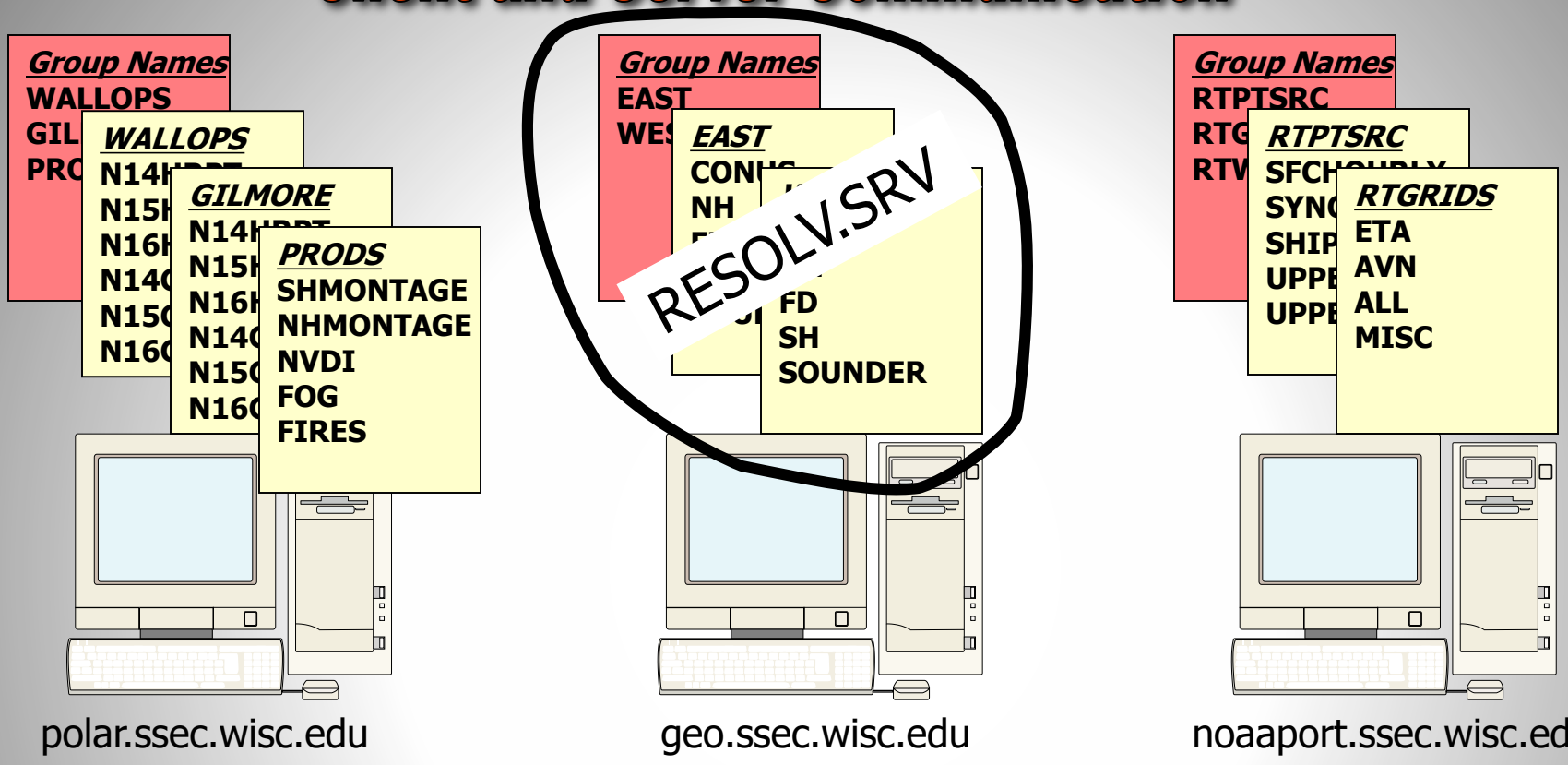
- Data file format converter
- Obey protocol for local and remote file access
- Features for efficient data transfer
- Features for integration into McIDAS-X and -V

Client/Server system structure

ADDE Client and Server



Client and Server Communication



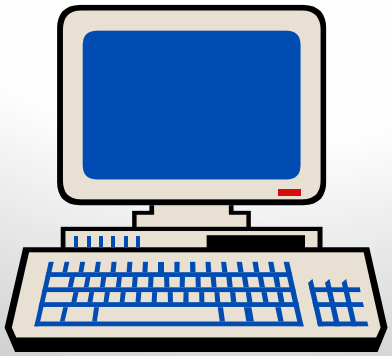
polar.ssec.wisc.edu

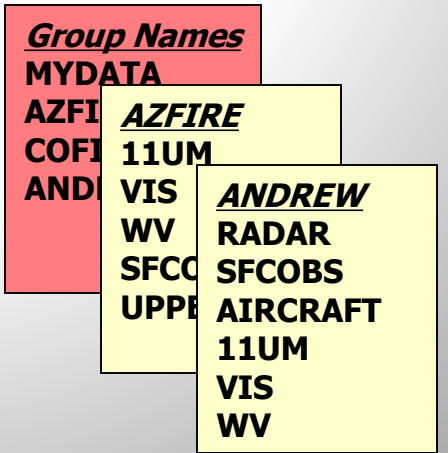
geo.ssec.wisc.edu

noaaport.ssec.wisc.edu

MCTABLE.TXT

Group	IP Address
WALLOPS	polar.ssec.wisc.edu
GILMORE	polar.ssec.wisc.edu
PRODS	polar.ssec.wisc.edu
EAST	geo.ss
WEST	g .edu
RTGRIDS	.ssec.wisc.edu
RTPT	port.ssec.wisc.edu
RTWA	noaaport.ssec.wisc.edu
AZFIR	local-data
ANDREW	local-data





ADDE Administrative Commands

Server Side

DSSERVE ADD *group/descriptor format bfile efile [keywords] "description*

Defines a dataset for the server workstation. Keywords may be required to add additional file name and location information. Group and descriptor are used by client commands to access datasets. Information stored in file named RESOLV.SRV.

Client Side

DATALOC ADD group ip_address

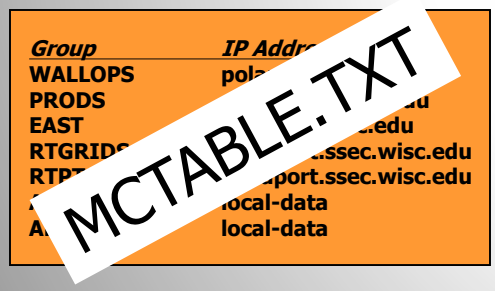
Creates a table in MCTABLE.TXT so client commands can determine which server workstation is to receive the ADDE request.

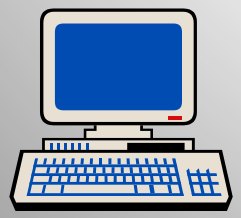
DSINFO type group

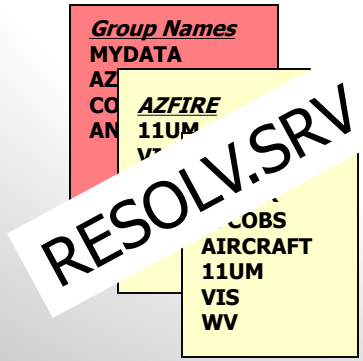
Lists descriptors for type and group specified.

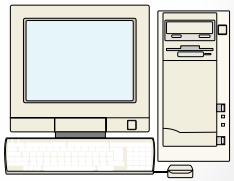
AKA alias group/descriptor

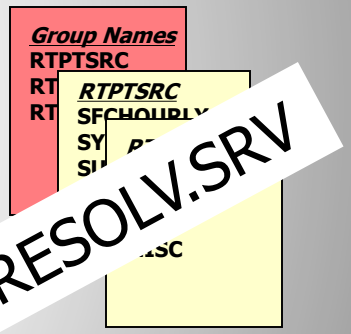
Creates an alias for a group/descriptor pairing.



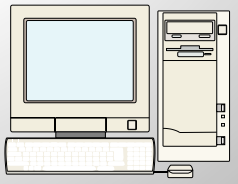


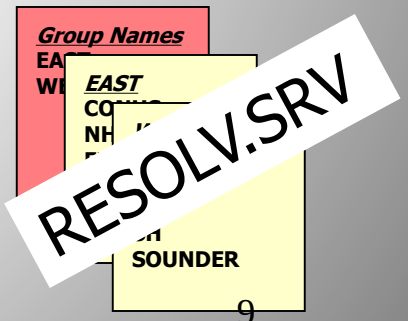






noaaport.ssec.wisc.edu



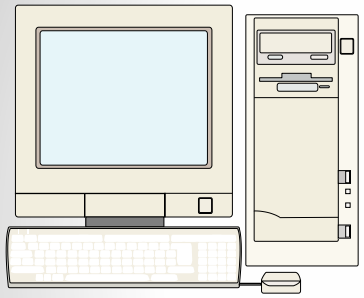


geo.ssec.wisc.edu

Client and Server Transaction

RESOLV.SRV

Dataset Names	*File Type*	*Location*
EAST/CONUS	SDI ingestor	filemask
EAST/SOUNDER	McIDAS Area	1-100
WEST/NH	SDI ingestor	filemask
WEST/SOUNDER	McIDAS Area	101-200



geo.ssec.wisc.edu

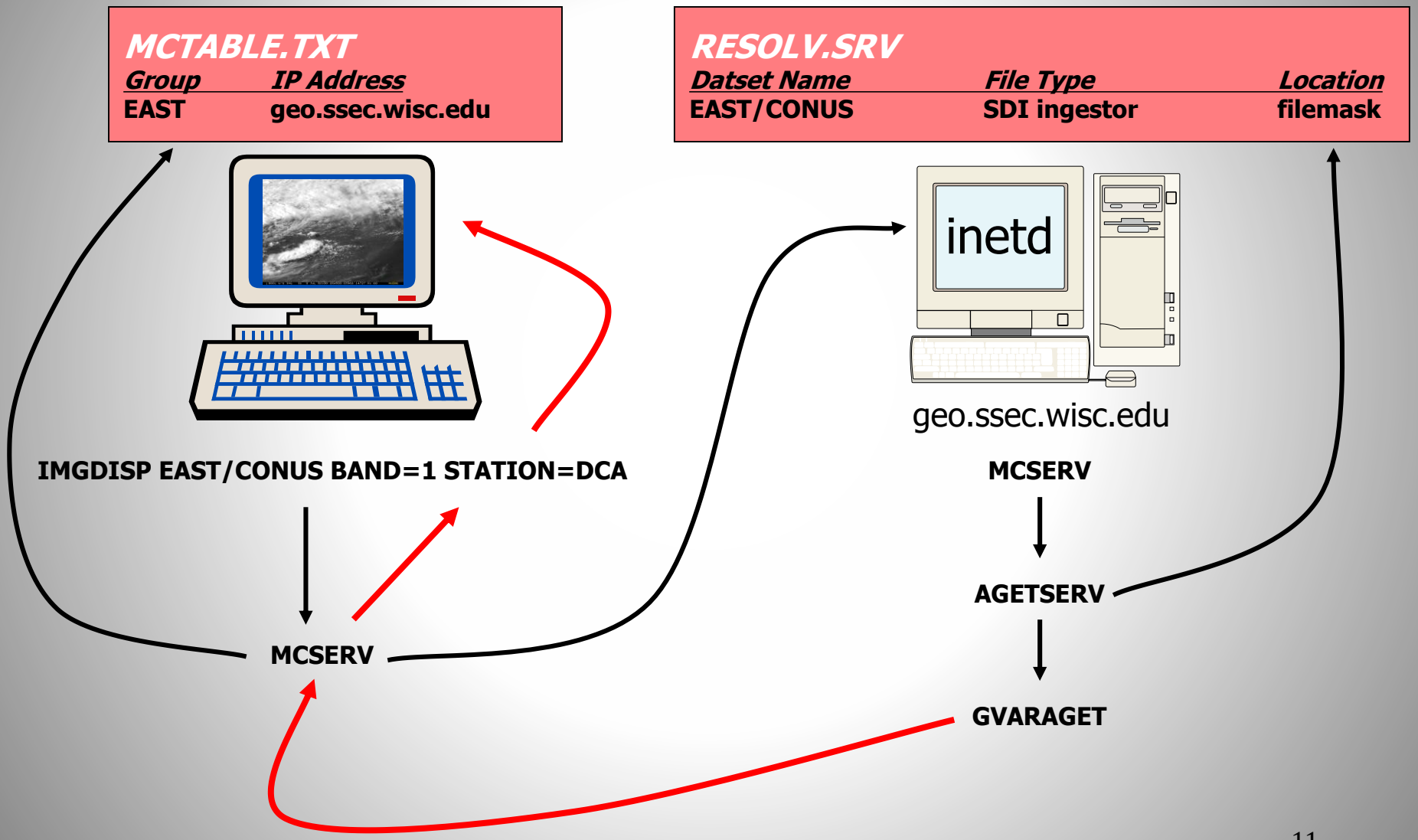
MCTABLE.TXT

Group	*IP Address*
WALLOPS	polar.ssec.wisc.edu
GILMORE	polar.ssec.wisc.edu
PRODS	polar.ssec.wisc.edu
EAST	geo.ssec.wisc.edu
WEST	geo.ssec.wisc.edu
RTGRIDS	noaaport.ssec.wisc.edu
RTPTSRC	noaaport.ssec.wisc.edu
RTWXTEXT	noaaport.ssec.wisc.edu
AZFIRE	local-data
ANDREW	local-data

EAST
CONUS
BAND 1
LOCATION DCA

IMGDISP EAST/CONUS BAND=1 STATION=DCA

Data Flow


MCTABLE.TXT
Group
IP Address
EAST
geo.ssec.wisc.edu
RESOLV.SRV
Datset Name
File Type
Location
EAST/CONUS
SDI ingestor
filemask
inetd
geo.ssec.wisc.edu
IMGDISP EAST/CONUS BAND=1 STATION=DCA
MCSERV
MCSERV
AGETSERV
GVARAGET

ADDE
Under the Hood

Mechanism

- The client requests a connection to a server
- The request causes the creation of a *pipe*, a *fork*, and the *exec* of the ADDE communications module, **mcserv**

Mechanism

- The client transmits over the *pipe*, and then receives on it
- A 16-byte preamble is sent from the client:
  - version number of the protocol: **0x0001**
  - IP address of the server machine
  - port number: **112**
  - service name; for example:

    aget

Mechanism

- **mcserv** examines the server address
  - Local: **mcserv** *exec*s a server, based on the service name. This server inherits the *pipe*, and does all further communication with the client.
  - Remote: **mcserv** continues, and acts as a *pipe* extender, using *TCP/IP* to the remote system. It next reads the 160-byte request block.

Request Block

Request header components	Length, in bytes	Word number in Fortran	servacct structure names in C
server IP address	4	1	server_address
server port used	4	2	server_port
client IP address	4	3	client_address
user ID	4 (ASCII)	4	user
project number	4	5	project
password	12 (ASCII)	6 - 8	password
transaction type	4 (ASCII)	9	transaction
number of bytes received by server	4	10	input_length
ADDE request string	120 (ASCII)	11 - 40	text

Mechanism

- If **mcserv** succeeds in connecting to the *port*:
  - Sends a resynthesized 16-byte preamble and 160-byte request block to the server
  - **mcserv** then reads and sends the number of bytes stored in the input data length field
- All information has been sent to the server, **mcserv** then:
  - Continues as an intermediary between the client application and the server
  - Copies bytes sent by the server to the *pipe* being read by the application.

Mechanism

- On the remote server machine:
  - a **mcserv** is started by **inetd**
  - Same steps are followed, except now the service is local
  - **mcserv** *exec*s the server based on the service name, etc.
  - When the server is finished sending its response, it sends a 92-byte trailer block, and exits.

Notes

- The design is stream oriented, so both the client and the server can be working simultaneously
  - The server locates the data and transmits it to the client via a *pipe* and/or *TCP/IP*
  - The client reads the *pipe* and operates on the data
  - The *pipe* is a finite size:
    - The server will wait to write if the *pipe* is full
    - The client will wait to read if the pipe is empty

Notes

- By default, if 120 seconds elapse with no activity on the pipe, the process stops. The process on the other end of the pipe also stops at this time.
  - The ADDETIMEOUT environment variable will adjust the timeout if more or less time is appropriate for this dataset

https://www.ssec.wisc.edu/mcidas/doc/users_guide/current/intro-34.html

McIDAS Area File Structure

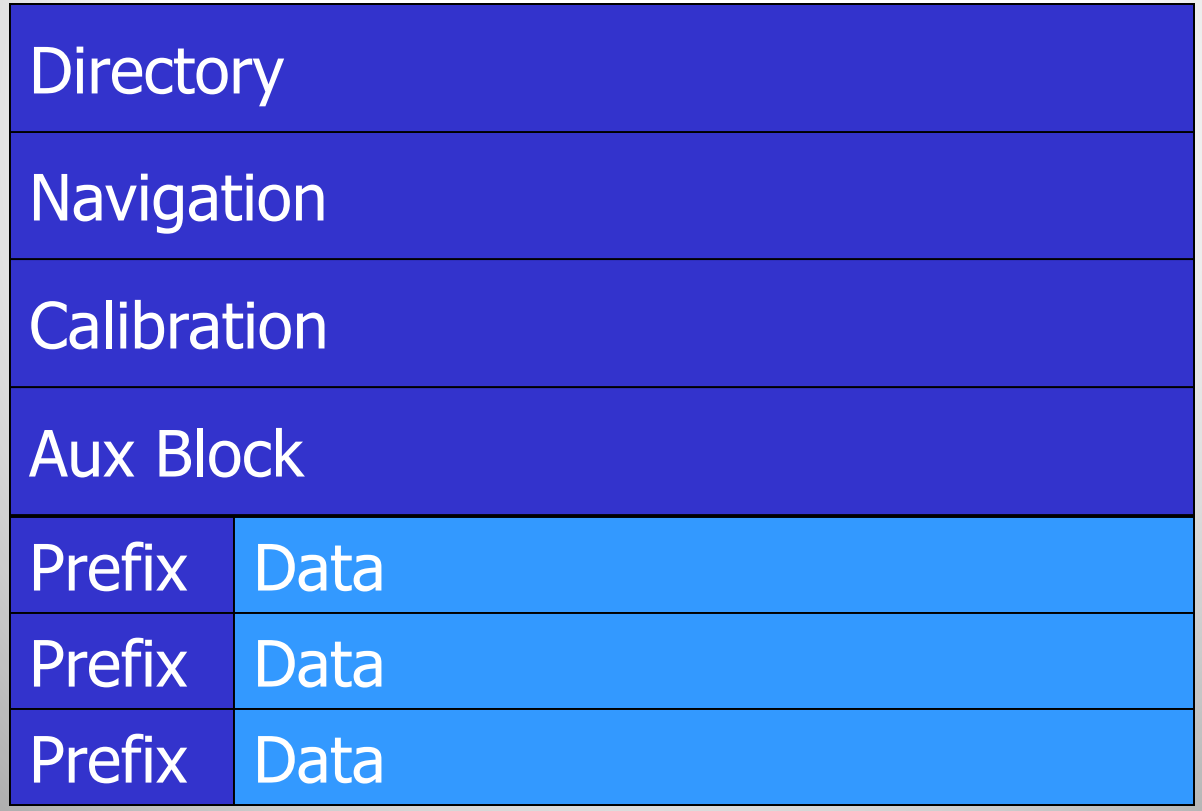
Satellite Image Data

Terminology

AREA####	**McIDAS file that stores image data (where #### is a value from 0001-9999)**
Directory	**First part of the AREA file which describes the image data**
Navigation	**Information necessary to co-locate satellite data to positions on earth**
Calibration	**Information necessary to convert raw counts into meaningful physical quantities**
Aux Block	**Auxiliary Information such as LALO navigation**
Prefix	**Header information for each line of data**

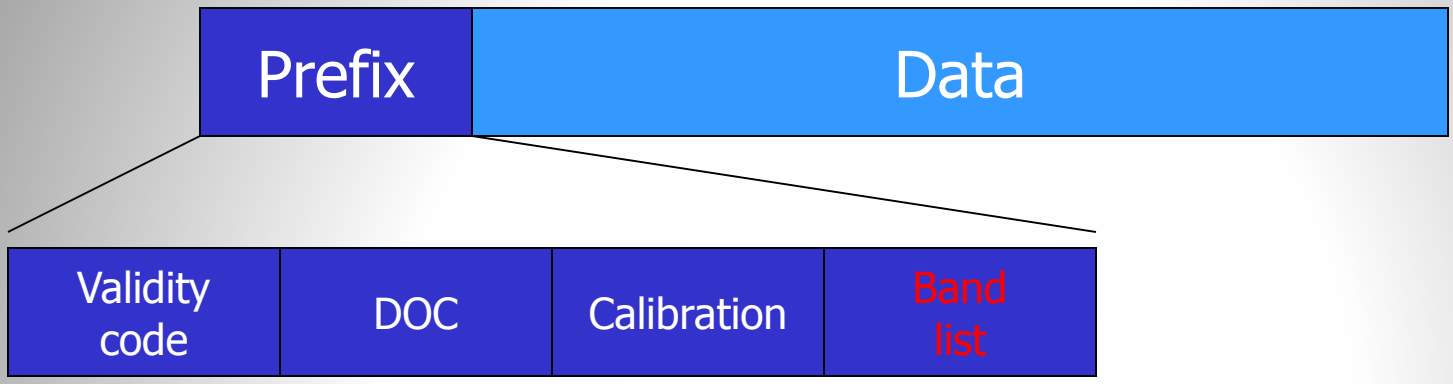
File Structure

AREA1234



Satellite Image Data

Line Prefix



Validity code(valcode)	A number that is used to determine if there is data for the line. This value is also stored in the area directory. Applications compare the valcode on the line with the valcode in the area directory. The line of data is considered missing if they do not match.
DOC	Satellite specific documentation.
Calibration	Coefficients required if calibration changes line by line.
Band list	Ordered list of bands found on each line.

Calibration

AREA Directory

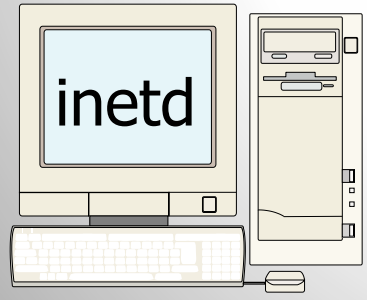
```
Image file directory listing for:EASTL/CONUS
 Pos Satellite/          Date        Time      Center      Res (km)    Image_Size
     sensor                                  Lat  Lon    Lat   Lon
 --- ------------  -----------  --------  ---- ----  ----- ----- ------------
 157  G-8 IMG        18 JUL 02199  18:45:00    22   71
    Band: 1   0.65 um Visible - Cloud Cover                1.13  0.58  4988 x13852
    Band: 2   3.9 um Night clouds; shortwave window        4.53  2.33  1247 x 3463
    Band: 3   6.8 um Upper level water vapor               4.53  2.33  1247 x 3463
    Band: 4  10.7 um Surface temp; longwave window         4.53  2.33  1247 x 3463
    Band: 5  12.0 um Sea surface temp and water vapor      4.53  2.33  1247 x 3463
     proj:    0 created: 2002199 184515  memo: RT GVAR
     type:GVAR     cal type:RAW
     offsets:  data=    3328 Calibration=  256 Navigation= 2816 Auxiliary=    0
     doc length: 228   cal length:   0   lev length:   0 PREFIX= 232
     valcod:  199184500 zcor:  0 avg-smp: N
     start yyddd: 2002199  start time:184515  start scan:  401
     lcor: 3205  ecor:  9049  bytes per pixel: 2  ss: 70
     Resolution Factors (base=1):   Line=    4.0   Element=    4.0
```

Calibration

IMGDISP EAST/CONUS BAND=4 STATION=DCA

BRIT
Temperature
Radiance

BRIT
Temperature
Radiance



geo.ssec.wisc.edu

Raw Counts

Raw Counts

Calibration Modules

GMS | GVAR | MSAT | AVHR

Radiance Values
albedo

Inverse Planck Equation or LUT

Temperatures

Standard GOES Temperature to Grayscale Conversion

Grayscale Values (BRIT)

Writing an ADDE Server

Two servers required

- Directory server returns
  - Area directory
  - Ancillary information
- Data server returns
  - Data as Area format (including directory)

Directory server

- Area directory
- Ancillary information
  - Center point (latitude/longitude) of data
  - Resolution (km) at center point
  - Possible units for band requested

Data server: File converter
Image data

- Reformat input file into a representation (Image Object) of a McIDAS-X Area file
- ADDE protocol returns Image Object to client application
- McIDAS-X application-compatible

ADDE protocol

- Data delivery protocol (via TCP/IP) developed at SSEC in the mid-1990s
- Protocols for image, grid, point, and text data
- Local and remote data access are handled the same

Flexible and efficient data access

1. Region of interest specification
2. Spectral subsecting
3. Unit conversion
4. Other features

1. Region of interest specification

- Center point of region of interest can be specified by:
  - File row, column
  - Image line, element
  - Geographic latitude, longitude
- Spatial subsecting (number of lines and elements)
- Sampling of lines and elements

2. Spectral subsecting

- For multi-banded data, return:
  - Single band (required)
  - All bands (if possible)
  - Multi-bands (optional)

3. Unit conversion

- Convention: By default, retain data values as stored in file (RAW values)
- At a minimum, a 1-byte grayscale value must be returned (BRIT)

3. Unit conversion

- Convert to useful units as requested by user, typically:
  - Infrared satellite data: conversions to radiance (RAD) and brightness temperature (TEMP)
  - Visible satellite data: conversions to reflectance (REF) or albedo (ALB)
- Other units can be defined, which are incorporated into a calibration module

4. Other features

- Return data values as 1-, 2-, or 4-byte (spacing)
  - Return in the most compact format
  - If data can not be returned in requested spacing, return error
- If possible, return values as unsigned integers

Navigation and Calibration modules

Navigation Modules

- Subsystem for adding new navigation type
- Each navigation module is a collection of four functions:
  - Defined API
  - Recollecting McIDAS-X is all that is needed

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/access-6.html#30949

Navigation Modules

- Naming convention, for example:
  - GOES-R ABI ABIN nvxabin.dlm
  - Rectilinear RECT nvxrect.dlm
- Four required functions:
  - NVXINI Initialization
  - NVXEAS Earth to satellite coordinate transform
  - NVXSAE Satellite coordinate to Earth transform
  - NVXOPT Optional transforms (e.g., satellite subpoint)

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/access-6.html#30949

Navigation Modules

- Writing in Fortran recommended
- Pre-compiler step (*convdlm*) renames the common functions to unique names, for example for ABIN:
    - NVXINI => NV1INIABIN
  - Generates source file *nvprep.f*

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/access-6.html#30949

Navigation Modules

- Consider using LALO navigation
  - Latitude/longitude point every n^{th} line and element
  - Good for low-volume images (< 2,000,000 points) or when sampling lat/lon values is possible (approximately linear variation within the sampled box)

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/formats-13a.html#LALO

Calibration Modules

- Subsystem for adding new calibration type
- Each calibration module is a collection of three functions:
  - Defined API
  - Recollecting McIDAS-X is all that is needed

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/access-7.html#23725

Calibration Modules

- Naming convention, for example:
  - GOES-R ABI ABI kbxabi.dlm
  - Product PRD kbxprd.dlm
- Three required functions:
  - KBXINI Initialization
  - KBXCAL Handles unit conversions (e.g., RAW=>TEMP)
  - KBXOPT Returns valid calibration types

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/access-7.html#23725

Calibration Modules

- Writing in Fortran recommended
- Pre-compiler step (*convdlm*) which renames the common functions to unique names, for example for ABI:
  - KBXINI => KB1INIABI
  - Generates source file *kbprep.f*

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/access-7.html#23725

ADDE Server Breakdown

Russ Dengel and Dave Santek

Space Science and Engineering Center

2016 McIDAS Users' Group Meeting
Madison,WI
14 November 2016

Goals of Course

- How to approach building an ADDE server
  - Converter vs. ADDE server
  - Data format issues
  - Navigation decisions
  - Calibration decisions
  - ADDE required features
- This course addresses only ‘image’ data

Where to begin?

Why write an ADDE server?

- Incorporate a new data type into McIDAS
  - Reads the native file format
  - No need to convert all your data into Area files, the conversion is done on-the-fly
  - Easy to provide the ADDE server to someone hosting the data for McIDAS users
- Disadvantage:
  - More difficult than a converter, because of the ADDE features and functionality

What about converters?

- IMGMAKE (-XRD)
  - Convert raster files (along with latitude/longitude arrays) to McIDAS Area files.
  - Supports:
    - Multi-band data
    - 1- or 2-byte data
    - TIFF and ENVI input
    - LUT for 1-byte calibration; PRD calibration
    - McIDAS-X geographic projections, polar (TBUS)

What about converters?

- TXT2MD (Core)
  - Convert point data in a text file to McIDAS MD format

Writing an ADDE server

- First, know your data:
  - File format API
  - How the data are geolocated (navigated)
  - How the stored values are converted to useful units
- Essentially, you need the knowledge to reshape the data into a McIDAS Area file
- All of the above are also required to write a ‘converter’

Data issues

- Are the files in a format using libraries already included in McIDAS-X?
  - netCDF, HDF-4, HDF-5
- If so, it's advisable to use an existing ADDE server as a model for the new one
- If not, still consider using an existing server as a template for the structure of the server
- If the data are not in a raster form, an ADDE image server may not appropriate

Navigation decisions

- Are the data in a geographic projection that is in the McIDAS-X library?
  - Lambert conformal, Mercator, polar stereographic, radar, rectilinear (lat/lon grid)
- If so, it's likely that the projection parameters are expressed differently (except rectilinear)
  - This may not be a trivial conversion

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/formats-13a.html

Navigation decisions

- Are latitude/longitude points included with the data?
  - If so, use the LALO navigation if there are less than 2,000,000 points in the image or the lat/lon points can be sampled to that amount
  - See supplementary doc included with this course (LALOBlockDoc.pdf) and the link below

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/formats-13a.html#LALO

Navigation decisions

- Is the image data in a satellite swath format?
  - Are only satellite orbit and instrument scanning information available?
  - How does the data provider expect users to determine lat/lon for data points?
  - There is no generic geostationary or polar orbiting navigation module in McIDAS-X
    - TBUS, TLE, rectified geo options

Navigation decisions

- Example navigation modules can be found in the McIDAS-X source directory: *nvx*.dlm*

Calibration decisions

- How are stored values converted to useful units, for example raw counts to brightness temperatures?
  - Is it through a lookup table?
  - Is there an equation?
  - Are the coefficients constant for the entire image or do they vary line-by-line?
  - Is it a simple conversion with no need for high precision?

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/access-7.html#23725

Calibration: Lookup table

- This is practical for tables of up to 32768 values (16-bit unsigned integers)
- If the lookup table is larger:
  - Consider fitting a polynomial to the table values, and storing the coefficients in the calibration block
  - Store the table in an external file (not recommended)

Calibration: Equation

- There are two ways to calculate using an equation:
  - Compute by pixel-by-pixel
  - Generate lookup table to reduce the amount of calculations
  - There are many byte-level and lookup table utilities in the McIDAS-X library (see link below)

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-5.html

Calibration: Constant coefficients

- Use the calibration block in Area file structure
- Store a set of coefficients for each band to handle single and multi-band Area files
- Use scaled integers for the coefficients as floating point numbers can get inadvertently byte-swapped during ADDE transfers.
  - Occurs when value represents valid ASCII characters

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/formats-1.html#16227

Calibration: Line-by-line coefficients

- Use the Calibration section in the line prefix to store the coefficients
- Include a known value in this Calibration section to handle big- and little-endian formatted Area files

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/formats-1.html#12392

Calibration: Simple conversion

- Consider using the product calibration (PRD) when the conversion is linear, low precision.
  - One-byte data stored in Area file
  - Calibration coefficients convert one-byte to useful values
  - See link below for the McIDAS-X PRDUTIL command
- Nice feature of different output units based on range of data values

http://www.ssec.wisc.edu/mcidas/doc/users_guide/current/prdutil.html

Calibration decisions

- Example calibration modules can be found in the McIDAS-X source directory: *kbx*.dlm*

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/access-7.html#23725

Additional

- Two data files may need to be modified in the ~mcidas/data directory:
  - If necessary, add a new McIDAS satellite source (SS) to SATANNOT
  - Add instrument/band list entry to SATBAND
- Check with the MUG for an SS number to use and how to edit these files

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/formats-28.html
http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/formats-27.html

Minimum required features

- Region selection:
  - Convert lat/lon to line/element
  - Extract rectangle of data centered on line/element
- Construct arrays of latitude and longitude for every n^{th} line/element
- Send data back as 1-byte grayscale values

ADDE Server Steps

1. Read the ADDE client request
2. Read the server mapping table
3. Interpret the client request
4. Retrieve requested data from disk
5. Send the data to the client
6. End the transaction

1. Read the ADDE client request

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/servers-3.html#42191

- See section: **Using a secondary server**
- Call **M0InitLocalServer** to read client request
- This gets the request block into a C structure

2. Read the server mapping table

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/servers-3.html#11020

- **M0sxdatasetinfo** reads RESOLV.SRV
  - RESOLV.SRV is written by DSSERVE command
- Contents of RESOLV.SRV are parsed and returned in a list of variables
  - Information on the dataset (e.g. directory/file mask)

3. Interpret the client request

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/servers-3.html#16284

- Retrieve parameters from the request string using **Mccmdstr**, **Mccmdint**, etc.
  - These are standard McIDAS-X command line argument fetchers
  - The parameters contain size, band, center point, etc. specifications

Image directory request syntax

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/servers-5.html#25171

Image data request syntax

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/servers-5.html#41467

4. Retrieve requested data from disk

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/servers-3.html#28039

You, the author of the server, must be familiar with the data file format, libraries, and APIs to read the data

5. Send the data to the client

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/servers-3.html#16225

- Send data to client using **M0sxsend**
- Data must be in big-endian format (network byte order)
  - Use **swbyt4** and **swbyt2** to switch bytes (no effect on big-endian machines)

6. End the transaction

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/servers-3.html#34069

- End transaction by calling **M0sxdone**
- Set appropriate return code
  - See ~mcidas/data/ADDERROR.DOC

An ADDE Directory Server

- Constructs a McIDAS directory block
- May return multiple directory blocks
- Returns
  - ADDE absolute position
  - Directory block
  - AUX cards

Directory: Transaction Parameters

- DAY: range of image dates
- TIME: range of image times
- BAND: range of image bands
- SS: range of Satellite IDs
- AUX: AUXiliary Cards
  - Calibration information
  - Basic geolocation information

Directory: Server Operations

- Process the transaction request parameters
- Make a list of files matching the ADDE file mask
- Decode the Day/Time info
- Time order the images

Directory: Server Operations

- Validate the image using request parameters
- Extract the dimensions, lat/lon, and datasets
- Construct the directory block
- Construct AUX cards
- Send the directory to the client

Directory: AUX cards

- Part of every directory transaction
- Information NOT contained in the directory block
  - Lat/Lon center point
  - Lat/Lon resolution at center point
  - Nominal Lat/Lon resolution
  - Calibrations per band

Directory: General Rules

- Directory Server will return a 65 word block (position + 64 word directory)
- DATE is stored in IYD format instead of CYD (done for Y2K)
  - IYD – *yyyddd*, where *yyy* is the number of years since 1900
- Band Map (word 18) is a bit map of bands contained in the source image

Some Rules: Byte Order

- ADDE servers ONLY send big-endian byte order
- ADDE servers are responsible for switch from little-endian to big-endian
- Client will handle transform to native byte order
- Some parts of the return transaction may contain ASCII text (not switched)

Some Rules: Resolution

- McIDAS uses a relative resolution instead of actual resolution
- 1-based where 1 is the highest resolution produced by the instrument
- Integer values only
- AUX cards will contain the ground based lat/lon resolution at subpoint

An ADDE Data Server

- Constructs an ADDE “image object”
- Returns only 1 image
- May return multiple image bands
- Includes navigation and calibration
- Image as described by the request

Data: Transaction Parameters

- pos: ADDE absolute position number
- DAY: day range to search
- TIME: time range to search
- place: request sector reference
- LMAG: line magnification
- EMAG: element manification

Data: Transaction Parameters

- size: line and element size
- BAND: requested image band(s)
- SPACE: bytes per element (1,2 or 4)
- UNIT: requested output units

Data: Server Operations

- Validate image based on the request
- Extract the Lat/Lon and data fields
- Subsect the data based on request coordinate system (Earth, Image or File)
- Subsect the Lat/Lon based on the request
- Sample data to requested resolution reduction

Data: General Rules

- The 4 byte law: All parts of an image line (prefix + data) sizes must be a multiple of 4 bytes
- Data is scaled integers only
- No negative numbers

Data: Calibration Rules

- RAW is the format of the data closest to the “raw” signal (file format)
- RAD is radiance; typically 4-byte
- TEMP is Kelvin; typically 2-byte
- BRIT is brightness; always 1-byte
  - GRYSCL function converts TEMP to BRIT
  - There is always a BRIT calibration

Debugging a Server

- Insert m0sxtrce calls into the source code
- Append TRACE=1 to McIDAS command to signal server to log debug messages
- ‘trce’ file will be created in:
  - mcidas/data directory running remote server
  - Current directory running local server
- IBM Rational Purify run-time debugger

Examine Server Code

- *scatadir.pdf*
- *scataget.pdf*
- Other ADDE servers of interest

McIDAS-X Programming Concepts

Dave Santek, Russ Dengel, Rick Kohrs

Space Science and Engineering Center

2016 McIDAS Users' Group Meeting
Madison, WI
14 November 2016

Goals of course

- To understand how to write a McIDAS-X application
  - Developer's overview
  - Structure of an application
  - Summary of available functions
  - Examine a basic application

Developer's overview

Source file naming convention

Suffix	Language	Description
`.c`	C	functions and McIDAS-X commands
`.cp`	C	ADDE servers and non-McIDAS-X applications
`.dlm`	Fortran	dynamic link modules
`.for`	Fortran	functions and subroutines
`.fp`	Fortran	ADDE servers and non-McIDAS-X applications
`.h`	C	include files
`.inc`	Fortran	include files
`.mac`	Fortran	McIDAS-X macros
`.pgm`	Fortran	McIDAS-X commands

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/basics-3.html

Developer's overview

Fortran: Hello World

```
subroutine main0
call sdest ('Hello World',0)
call mccodeset (0)
return
end
```

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/basics-3.html

Developer's overview

C: Hello World

```
#include <stdio.h>
#include "mcidas.h"

int main (int argc, char **argv)
{

  /* initialize the McIDAS environment */

  if (Mcinit
 (argc, argv) < 0)
  {
    fprintf  (stderr, "%s\n", Mciniterr ());
    return (1);
  }

  Mcprintf ("Hello World\n");

  Mccodeset (0);
  return (Mccodeget());
}
```

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/basics-3.html

Developer's overview

Fortran: Help section

```
C ? NAME -- Describe the purpose of this command
C ?    NAME FUNCT1 parm1 parm2 <keywords> "quote
C ?    NAME FUNCT2 parm1 <keywords>
C ? Parameters:
C ?    FUNCT1    | describe the purpose of this function option
C ?    FUNCT2    | describe the purpose of this function option
C ?    parm1     | describe this parameter (def=default value)
C ?    parm2     | describe this parameter (def=default value)
C ?    "quote    | describe the contents of the quote string
C ? Keywords:
C ?    KEYNAME=  | describe values (def=default values)
C ?    KEY2=YES  | describe effect (def=default value)
C ? Remarks:
C ?    Add remarks, from most to least important. Use complete
C ? sentences. Separate multiple remarks with a single blank line,
C ? as below.
C ?
C ?    Always end the help section with a line of 10 dashes,
C ? as below.
C ? ----------
```

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/basics-3.html

Developer's overview

C: Help section

```
/*
*?  NAME -- Describe the purpose of this command
*?     NAME FUNCT1 parm1 parm2 <keywords> "quote
*?     NAME FUNCT2 parm1 <keywords>
*?  Parameters:
*?     FUNCT1    | describe the purpose of this function option
*?     FUNCT2    | describe the purpose of this function option
*?     parm1     | describe this parameter (def=default value)
*?     parm2     | describe this parameter (def=default value)
*?     "quote    | describe the contents of the quote string
*?  Keywords:
*?     KEYNAME=  | describe values (def=default values)
*?     KEY2=YES  | describe effect (def=default value)
*?  Remarks:
*?     Add remarks, from most to least important. Use complete
*?  sentences. Separate multiple remarks with a single blank line,
*?  as below.
*?
*?     Always end the help section with a line of 10 dashes,
*?  as below.
*?  ----------
*/
```

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/basics-3.html

Mixed language programming

- McIDAS contains Fortran and C library routines:
  - In many cases, Fortran can call C routines directly and vice versa.
  - For some routines, there are C or Fortran jackets to make it easy (especially, when strings are being passed)

Mixed language programming

Fortran calling Fortran

```
integer     tvlin, tvele
integer onscreen
real    lat, lon

tvlin   = 100
tvele   = 200

onscreen    = iyxll(tvlin, tvele, lat, lon)
```

C calling Fortran

```
#include "mcidas.h"

Fint iyxll_ (Fint *, Fint *, Freal *, Freal *); /* function prototype */

Fint    tvlin, tvele;
Freal   lat, lon;
Fint    onscreen;

tvlin   = 100;
tvele   = 200;

onscreen= iyxll_ (& tvlin, & tvele, & lat, & lon);
```

Developer's overview

- The link below has other basic information, including:
  - Suggestions on debugging
  - Programming do's and don'ts

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/basics-3.html

Simple Fortran program

```
 subroutine main0

       implicit none

       integer        len_trim
       integer        mccmdint
       integer        mccmddbl
       integer        mccmdstr

C ---    local variables

       integer        iret
       integer        ival
       double precision dval
       character*80    cstr
       character*80    print
```

Maximum allowed value

Minimum allowed Value

Default Value

```
      iret = mccmdint(' ',1,'Integer Value',15,0,9000,ival)
      if (iret .lt. 0) goto 2000
      iret = mccmddbl(' ',2,'Double Precision Value',
     &            15.,0.,9000.,dval)
      if (iret .lt. 0) goto 2000
      iret = mccmdstr(' ',3,'String Value',cstr)
      if (iret .lt. 0) goto 2000

      call sdest(' integer value is: ',ival)
      write(print,*) ' double value is ',dval
      call sdest(print,0)
      call sdest(' String value is >'//cstr//'<',0)
      call sdest(' String value is >'//cstr(1:len_trim(cstr))//'<',0)
2000  continue
       call sdest('TEST:  Done',0)
```

Simple C program

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "mcidas.h"
#include "mcidasp.h"
#include "m0arg.h"

int main (int argc, char **argv)  {

  int                 ok;
  int                 length;
  double                dlength;
  const char *          cstr;

  /* initialize mcidas environment */
  ok = Mcinit (argc, argv);
  if (ok < 0)    {
    fprintf (stderr, "%s\n", Mciniterr ());
    goto End_Of_Main;
  }

  ok = Mccmdint(" ",1,"Test Integer Value",0,1,5,&length);
  if (ok < 0) goto End_Of_Main;
  ok = Mccmddbl(" ",2,"Test Double Value",0.,1.,0.,&dlength);
  if (ok < 0) goto End_Of_Main;
  ok = Mccmdstr(" ",3,"Test Double Value",&cstr);
  if (ok < 0) goto End_Of_Main;

  Mcprintf("length read in as %d \n",length);
  Mcprintf("length read in as %f \n",dlength);
  Mcprintf("length read in as >%s< \n",cstr);


  End_Of_Main: Mcprintf("TEST:  Done\n");
  return(0);

}
```

Command line argument fetching

- mccmd* functions read in keywords and positional parameters from the command line
  - * = str, int, dbl, iyd, ihr, dhr, ill, dll, quo (different variable types)

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-2.html

Command line argument fetching

C function	Fortran function	Description
Mccmddbl	**mccmddbl**	extracts a value as a double precision number
Mccmddhr	**mccmddhr**	extracts a time value as a double precision number in units of hours
Mccmddll	**mccmddll**	extracts a latitude or longitude value as a double precision number in units of degrees
Mccmdihr	**mccmdihr**	extracts a time value as an integer value in the hhmmss format
Mccmdill	**mccmdill**	extracts a latitude or longitude value as an integer value in the dddmmss format
Mccmdint	**mccmdint**	extracts a value as an integer
Mccmdiyd	**mccmdiyd**	extracts a day value as an integer value in the Julian day format ccyyddd
Mccmdquo	**mccmdquo**	extracts a character string value from the quote field
Mccmdstr	**mccmdstr**	extracts a character string value

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-2.html

Command line argument fetching

```
rc = mccmdint('GRA.PHICS',1,'GRA Frame #',
                    mcgetgraphicsframenumber( ),1,
                    maxgraframe,ifrab)

rc = mccmdint(' ',2,'Graphics Frame Number',
                    mcgetgraphicsframenumber( ),1,
                    maxgraframe,ifrab)
```

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-2.html

User Common (UC)

- Blocks of shared memory
- Divided into 2 regions
  - Positive UC is constant across session
  - Negative UC is constant across application
  - Documented in ~mcidas/data/UC.DOC
- Contains both system- and user-level information

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-4.html

User Common (UC)

C function	Fortran function	Description
Mcluc	**luc**	returns a value from User Common
Mcpuc	**puc**	changes a value in User Common

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-4.html

User Common (UC)

C function	Fortran function	Description
McGetGraphicsFrameNumberI	**mcgetgraphicsframenumberi**	returns the current graphics frame number for an interactive application
McGetImageFrameNumberI	**mcgetimageframenumberi**	returns the current image frame number for an interactive application
McGetGraphicsFrameNumber	**mcgetgraphicsframenumber**	returns the current graphics frame number
McSetGraphicsFrameNumber	**mcsetgraphicsframenumber**	sets the current graphics frame number
McGetImageFrameNumber	**mcgetimageframenumber**	returns the current image frame
McSetImageFrameNumber	**mcsetimageframenumber**	sets the current image frame number

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-4.html

Error handling

C function	Fortran function	Description
Mccodeset	**mccodeset**	sets the global status to return upon exiting
Mccodeget	**mccodeget**	returns the current value of the global status
not available	**mcabort**	sends an error message and exits
Mciniterr	not available	returns a string explaining why Mcinit failed

Do:	Don't:
call **`edest`**, call **`mccodeset`**, and **`return`** a status in a function return code	call **`mcabort`** or **`exit`** since they are not very informative

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-4.html

Conversion utilities

1. Manipulating data at the byte level
2. Handling character strings
3. Converting day and time formats
4. Converting latitude and longitude formats
5. Converting physical units such as speed and temperature

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-5.html

1. Byte manipulation

movcw	copies the entire contents of a character string buffer to an integer buffer
movh	copies a number of half-words from a source buffer to a destination buffer with half-word increments
movpix	copies a number of elements from a source buffer to a destination buffer with incremental offsets for both the source and destination buffers
movw	copies a number of words from a source buffer to a destination buffer
movwc	copies the contents of an integer buffer to a character string buffer, copying as many bytes as can be stored in the destination string
mpixel	copies elements of a specified size in a source buffer to a destination buffer with elements of a specified size
mpixtb	like mpixel, but additionally converts variable-sized elements in a buffer based on a lookup table
pack	moves the least significant byte from each element of a word array and compresses it into consecutive bytes in a buffer

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-5.html

2. Character strings

Mcstrtodbl	**mcstrtodbl**	converts a character string in decimal point format to a double-precision value
Mcstrtodhr	**mcstrtodhr**	converts a character string in time format to a double- precision value
Mcstrtodll	**mcstrtodll**	converts a character string in lat/lon format to a double-precision value
Mcstrtohex	**mcstrtohex**	converts a character string in hexadecimal to an integer value
Mcstrtohms	**mcstrtohms**	converts a character string in time format to its components: hours, minutes, seconds
Mcstrtoihr	**mcstrtoihr**	converts a character string in time format to an integer value in units of hours/minutes/seconds
Mcstrtoill	**mcstrtoill**	converts a character string in lat/lon format to an integer value in units of degrees/minutes/seconds

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-5.html

3. Day and time conversions

Mccydtodmy	**mccydtodmy**	converts a Julian day in the form *ccyyddd* to day, month and year
Mccydtodow	**mccydtodow**	converts date to day of the week
Mccydtoiyd	**mccydtoiyd**	converts date in *ccyyddd* format to *yyyddd*
Mccydtostr	**mccydtostr**	converts the Julian day in the form *ccyyddd* to a variety of character string formats
Mccydtoyd	**mccydtoyd**	converts a 7-digit Julian day to 5 digits
Mcdaytimetosec	**mcdaytimetosec**	converts a Julian day in the form *ccyyddd* and the time of day in the form *hhmmss* to seconds since 1 January 1970 at 00 UTC
Mcdhrtoihr	not available	converts hours stored in double precision to hours stored in an integer of the form *hhmmss*
Mcdmytocyd	**mcdmytocyd**	converts day, month and year to a Julian day in the form *ccyyddd*
Mcgetday	**mcgetday**	gets the current system Julian day in the form *ccyyddd*
Mcgetdaytime	**mcgetdaytime**	gets the current Julian day in the form *ccyyddd* and the current time of day in the form *hhmmss*

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-5.html

4. Latitude/Longitude conversions

- **flalo**: converts an integer representation of latitude or longitude in the format *dddmmss* to a single-precision float, in degrees
- **ilalo**: converts a single-precision latitude or longitude to an integer value in the format *dddmmss*

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-5.html

5. Unit conversions

- **mcucvtd**: converts a list of double-precision values from one physical unit to a different physical unit
- **mcucvtr**: converts a list of single-precision values from one physical unit to a different physical unit

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-5.html

5. Unit conversions

Attribute	Valid units	Interface representation
length	meters	M
	kilometers	KM
	decameters	DM
	centimeters	CM
	millimeters	MM
	miles	MI
	nautical miles	NMI
	yards	YD
	feet	FT
	inches	IN
	degrees of latitude	DEGL
speed	miles per hour	MPH
	knots	KT or KTS
	meters per second	MPS
	feet per second	FPS
	kilometers per hour	KPH
temperature	Kelvin	K
	Fahrenheit	F
	Celsius	C

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-5.html

5. Unit conversions

pressure	millibars inches of Mercury pascals hectopascals	MB INHG PA HPA
time	hours minutes seconds days years	HR MIN SEC DAY YR
weight	grams kilograms pounds ounces tons	G KG LB OZ TON

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-5.html

Science utilities

McCape	**mccape**	computes Convective Available Potential Energy
McCoriolis	**mccorfor**	computes the coriolis parameter ($f=2\Omega\sin\Phi$)
McDewpt	**mcdewpt**	computes dewpoint
McDirec	**mcdirec**	computes meteorological direction of wind for u- and v-components
McDiver	**mcdiver**	computes divergence $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}$
McDivergeParm	**mcdivergeparm**	compute divergence of gridded parameter
McHeatIndex	**mcheatindex**	computes the heat index, given the temperature and dewpoint

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-6.html

Science utilities

McTheta	**mctheta**	potential temperature
McThetae	**mcthetae**	equivalent potential temperature
McThetaw	**mcthetaw**	wet bulb potential temperature
McUandV	**mcuandv**	computes wind u- and v-components
McVirtTemp	**mcvirttemp**	virtual temperature
McVort	**mcvort**	computes vorticity $\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$
McWetBulb	**mcwetbulb**	wet bulb temperature
McWindChill	**mcwindchill**	computes the wind chill, given the temperature and wind speed (old formula)
McWindChill2001	**mcwindchill2001**	computes the wind chill, given the temperature and wind speed (uses the 2001 algorithm)
not available	**rmix**	determines the mixing ratio, given the temperature and pressure

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-6.html

Text output

- C and Fortran routines

C function	Fortran function	Message type
Mcprintf	**sdest**	standard text
Mceprintf	**edest**	error
Mcdprintf	**ddest**	debug

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-3.html

Text output

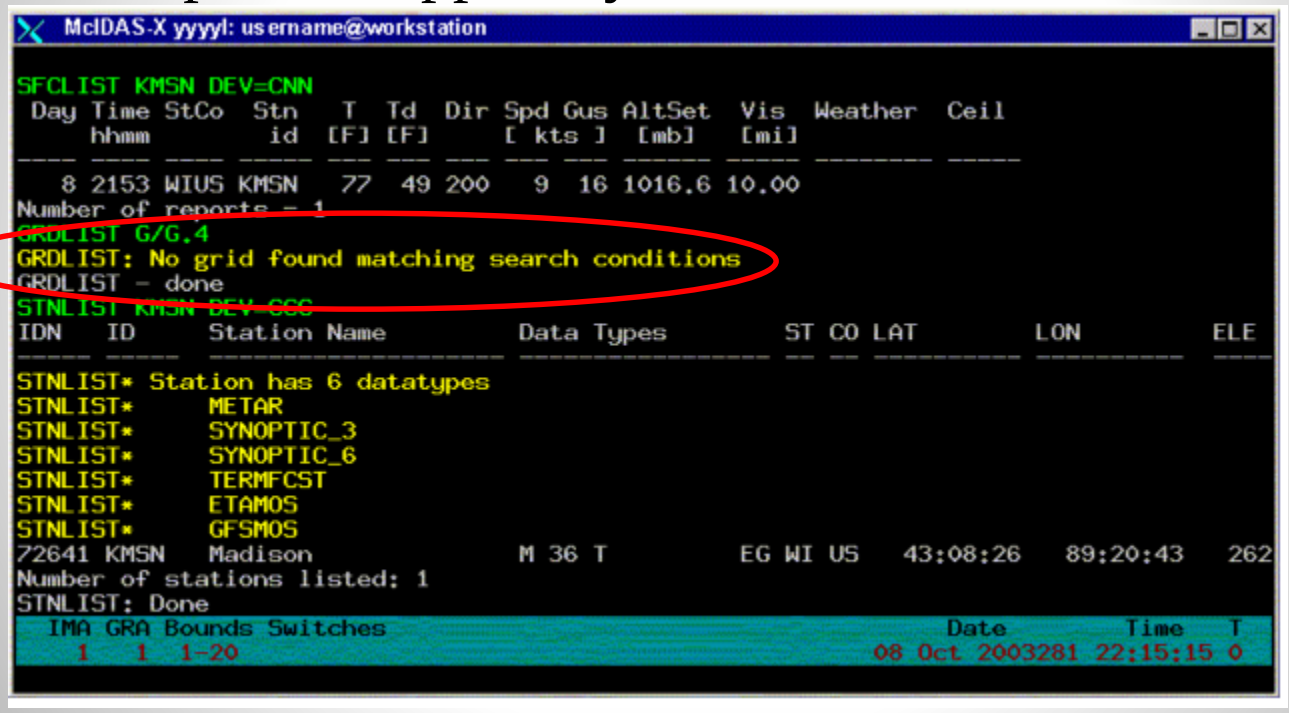
- sdest - standard messages

```
subroutine main0
call sdest ('Hello World',0)
call mccodeset (0)
return
end
```

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-3.html

Text output

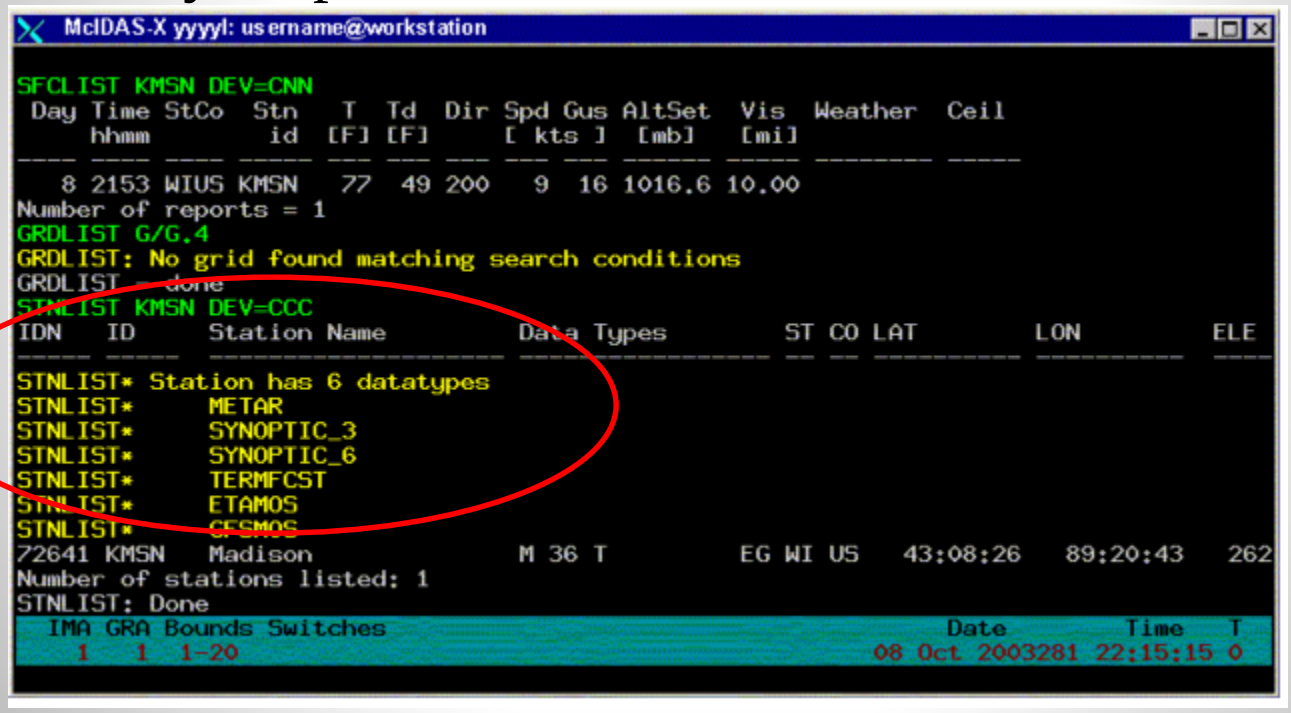
- edest - error messages
  - Output text appears yellow in text window



http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-3.html

Text output

- ddest - debug messages
  - Only output with DEV=CCC on command line



http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-3.html

Drawing graphics

Fortran function	Description
dshoff	turns the dash mode off
dshon	turns the dash mode on
endplt	closes or binds off the graphics frame
enpt	flushes the graphics buffer
initpl	initializes the graphics package to write to a frame object
newplt	erases the current graphics frame
page	resizes the viewport or world
plot	draws a line
pltdig	writes an integer number
qgdash	returns the current dash mode
qscale	returns the current scaling mode
sclhgt	converts text height from world to frame coordinates; use only if scaling is turned on; does not reduce the height of the text
scloff	turns scaling off; subsequent graphics calls use frame coordinates
sclon	turns scaling on; subsequent graphics calls use world coordinates
sclpnt	converts a point from world to frame coordinates
wrtext	writes a text string

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-3.html

Drawing graphics: Example

```
C --- Initialized variables

      integer       width   ! line width (session current)

      data      width  /   0/

C -------------------------------------------------------------
C INITIALIZE
C -------------------------------------------------------------

C     // Initialize the plot and fetch the color

      frame = luc(-1)
      call initpl( frame, width )
      if( mccmdint('COL.OR', 1, 'Graphic Color', 3, 1, 3,
     &  color ) .lt. 0 ) return
```

Set variables to the box corners, in screen coordinates

```
call box ( w_ullin, w_ulele, w_lrlin, w_lrele, color )
```

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-3.html

Drawing graphics: Example

```
      subroutine box( ullin, ulele, lrlin, lrele, color )


      implicit      NONE

---   interface variables

      integer       ullin
      integer       ulele
      integer       lrlin
      integer       lrele
      integer       color

---   symbolic constants and shared values

      integer       PEN_UP  ! move pen without drawing
      parameter             ( PEN_UP = 0)


      call plot( ullin, ulele, PEN_UP )
      call plot( ullin, lrele, color  )
      call plot( lrlin, lrele, color  )
      call plot( lrlin, ulele, color  )
      call plot( ullin, ulele, color  )
      call enpt

      return
      end
```

http://www.ssec.wisc.edu/mcidas/doc/prog_man/current/utilities-3.html

Example application

imgcheck.pgm

```
 9   C ? IMGCHECK -- Lists basic statistical values for an image
10   C ?    IMGCHECK dataset <keywords>
11   C ? Parameters:
12   C ?    dataset | ADDE dataset name and position; specify as alias.position or
13   C ?              group/descriptor.position; to use the default position,
14   C ?              either enter "0" or omit the .position portion
15   C ?              (no def for alias or group/descriptor, def=0 for position)
16   C ? Keywords:
17   C ? *********************  Keywords  ***********************
18   C ?    BINs= | number of histogram bins (def=16)
19   C ?    TAILs=min max | list percentages for values below min and above max
20   C ?                    (def=0 255)
21   C ? Remarks:
22   C ?    Statistics are limited to units of BRIT.
23   C ? ------------
24         subroutine main0
```

Parameter fetching

```
 97   c
 98   c ---  Get source dataset information
 99   c
100
101         status=mccmdstr(' ',1,' ',area)
```

```
303         status = mccmdint('TAIL.S',1,'Minimum Value',0,0,255,min_count)
304         status = mccmdint('TAIL.S',2,'Maximum Value',255,0,255,max_count)
305         status = mccmdint('BIN.S',1,'Number of Bins',16,1,256,num_bins)
306
```

Setup for data read

```
129
130   c
131   c --- For now we assume entire image
132   c
133         nsort = nsort + 1
134         sorts(nsort) = 'SIZE 99999 99999'
135
136   c
137   c --- add on the position to the sort clauses
138   c
139         nsort = nsort + 1
140         sorts(nsort) = 'POS ' // area_pos
```

```
150    c
151    c --- open the connections to source images
152    c
153          format = 'I1'
154          unit = 'BRIT'
155
156    c
157    c --- Get two handles
158    c
159          if (mcaget(area_name,nsort,sorts,unit,format,MAX_BYTE,1,
160         &            area_dir, area_handles(1)) .ne. 0) then
161             call mccodeset (2)
162             goto 9995
163          endif
164
```

Read data

```
319   c --- First time through image to calculate the average of all pixels
320   c --- Also generate the historgram statistics
321   c
322         do 200 line = begin_line,line_size
323
324   c
325   c --- Read in line of data
326   c
327             status = mcalin(input_handles(1), input_line)
328             if (status .ne. 0) then
329             call edest('Line Transfer error: line=',line)
330                 status = mcafree(input_handles(1))
331                 statistics = -1
332                 return
333             endif
334
335   c
336   c --- crack the data (if necessary)
337   c
338             if (element_bytes .ne. 4) then
339                 call mpixel(element_size,element_bytes,4,input_line)
340             endif
```

Operate on data

```
388
389               do 101 ele = 1,element_size
390                     input_count = input_line(ele)
391                     square_count = square_count +
392         &                 ((float(input_count) - pixel_mean))**2 /
393         &                   float(total_pixels)
394
395   101         continue
396   201     continue
397
398   c       pixel_mean =  float(total_count) / float(total_pixels)
399           standard_dev = square_count**.5
400
401           cyd = input_dir(4)
402
403           if (mccydok(cyd) .ne. 0) then
404               status = mciydtocyd(input_dir(4),cyd)
405           endif
```

Output statistics

```
422         out_string = ''
423         write(cpixel_mean,'(f11.1)') pixel_mean
424         out_string(1:10) = 'Average = '
425         out_string(11:22) = cpixel_mean
426         call bsquez(out_string)
427         call sdest(out_string,0)
428
```

Any questions or other topics?