DS-IRF-UM-0006 Issue: 2
Date: 1995 September 29 Rev.: 0

CSDS User Interface ISDAT User Defined IDL clients

Swedish Institute of Space Physics, Uppsala Division
S-75591 Uppsala, Sweden

with change bars for changes introduced in issue 2.0

Contents

1 Introduction

1.1 Intended readership

This manual is intended for the user of the CSDS User Interface ISDAT Client package who wants to design and add his personal IDL clients.

1.2 Applicability of the manual

The current version of the document applies to the ISDAT version 2.2, delivered as release 4 within the CSDS User Interface Project. It is valid for UNIX, SUN Solaris workstations.

1.3 Purpose of the software

The purpose of the CSDS User Interface Data Manipulation software package is to provide the scientific community with software tools to manipulate and display Cluster CSDS summary and primary parameters. The IDL provides a means to add user defined software modules for data manipulation and display.

1.4 How to use this document

This document basically consist of comments to the source code of a simple IDL client to be included in an ISDAT client package. The intention is that the reader from this should be able to make modifications and additions in order to meet his personal needs.

1.5 Related documents

An overview of the CSDS UI ISDAT Client Package is given in [Ref. 2]. It is assumed that the reader is familiar with the information given in that manual. The installation of the ISDAT client package is described in [Ref. 1]. Instructions for adding user defined clients written in C language are given in [Ref. 5]. The IDL is described in [Ref. 7].

1.6 Conventions and acronyms

In the following, we will use:

- *italics* to indicate exact names or expressions.
- Courier fonts to give command line expressions.
- > to indicate the terminal prompter.

Acronyms and abbrieviations used are described in Table 1.

Acronym	Meaning
CSDS	Cluster Science data System
CUI	CSDS User Interface
IDL	Interactive Data Language
IRF-U	Institutet för Rymdfysik, Uppsalaavdelningen
	Swedish Inst. of Space Phys., Uppsala Division
ISDAT	Interactive Science Data Analysis Tool
UI	User Interface

Table 1: Acronyms and abbrieviations

1.7 Reference Documents

[1] CSDS User Interface, ISDAT Installation Manual. Technical Report DS-IRF-IM-0001, IRF-U, September 1995.

[2] CSDS User Interface, ISDAT User Manual. Technical Report DS-IRF-UM-0001, IRF-U, September 1995.

[3] CSDS User Interface, ISDAT cuigr Client User Manual. Technical Report DS-IRF-UM-0003, IRF-U, September 1995.

[4] CSDS User Interface, ISDAT cuitm User Manual. Technical Report DS-IRF-UM-0004, IRF-U, September 1995.

[5] CSDS User Interface, Writing User Defined C Clients to be included in the ISDAT Client Package. Technical Report DS-IRF-UM-0007, IRF-U, September 1995.

[6] Reference document for CSDS CDF implementation. Technical Report DS-QMW-TN-0003, Queen Mary and Westfield College, October 1994. Issue 1.1.

[7] IDL reference guide. Technical report, Research Systems Inc., April 1994. Version 3.6.

[8] WECdata structure working group. Editor C. Harvey. The structure of the WEC/ISDAT data. Technical Report CWD-OBSPM-DD-001, OBSPM, March 1995.

1.8 Problem reporting

Problems should be reported to the CSDS National Data Centre.

2 Writing a simple ISDAT IDL client

2.1 Prerequisites and assumptions

For these instructions, it is assumed that you are familiar with IDL and the IDL syntax. If that is not the case, please consult your IDL manuals, for example [Ref. 7]. It is also assumed that your local work station has IDL properly installed.

In order to find the files without problems it is recommended to put the following lines in your personal .profile.

```
IDL_DIR = "your idl directory"; export IDL_DIR
IDL_PATH = +$IDL_DIR/lib:$CUI_PRD_ROOT; export IDL_PATH
```

2.2 IDL client cuit

We are now going to write a simple IDL client that is capable of displaying one specific CSDS parameter. We start by writing a main program:

2.2.1 Main program

First we have to set up three common blocks in order to make the variables available throughout the program:

```
common info, info
common spec, dspec
common database, db
```

The meaning of these will be explained later. Then we initialize the whole process by:

```
tmInfoFd = IdlInitialize()    ; must be executed first in program
!y.margin = [3.0, 3.0]        ; override the pre-set y-margins
window, 1                     ; create a window
```

IdlInitialize() is described in Appendix A. Here a connection is established with an active *time manager* It is important that this is done as the first executable statement in all IDL clients. We then set our preferred y-margins and create a window.

Now we can open a connection to an ISDAT server (data base handler):

```
db = IdlOpen()
if (db eq 0) then begin
    print, 'no database handler running'
    exit
endif
```

If, however, the client cannot find a running server, then we exit with an error message. *IdlOpen* is described in Appendix A.

Our IDL client will be a slave under a running *time manager* presumably *cuitm*, and we have to get information about the current time interval set by the time manager:

```
    info = GetTmInfo()
```

GetTmInfo() is described in Appendix A.

The next initializing step is to "hard code" the parameter to be plotted:

```
    name = ['CSDS_PP', 'C1', 'EFW', 'E_dusk', '', '', '']
    print, 'name=', name
    dspec = IdlName2Spec(db, name)
```

where the name string consist of the hierarchical ISDAT parameter description *project member, instrument, sensor, signal, channel, parameter* where in this example: *project* is CSDS_PP for the prime parameters, *member* is C1 for the first Cluster satellite, *instrument* is set to EFW for the EFW instrument etc. Valid names for *project, member, instrument* in the CSDS User Interface applications are found in [Ref. 3]. Valid names for the lower level entries in the hierarchy are found in [Ref. 6]. In this example, the three lowest levels are not used. The print statement above merely produces a printout in the terminal window. The third statement formats the parameter specification to a format understandable for the ISDAT server. *IdlName2Spec()* is described in Appendix A.

Finally, after having completed the initializations, we enter the main loop pending callbacks from the time manager, and we remain in the main loop until we exit the program. The main loop is described in section 2.2.2.

```
    MainLoop, tmInfoFd
```

Our complete main program is the following

```
; main program

    common info, info
    common spec, dspec
    common database, db

    tmInfoFd = IdlInitialize()       ; must be executed first in program
    !y.margin = [3.0, 3.0]
    window, 1

    db = IdlOpen()                   ; Open connection to data base handler.
    if (db eq 0) then begin
        print, 'no database handler running'
        exit
    endif

    info = GetTmInfo()

    name = ['CSDS_PP', 'C1', 'EFW', 'E_dusk', '', '', '']
    print, 'name=', name
    dspec = IdlName2Spec(db, name)

    MainLoop, tmInfoFd
end
```

2.2.2 The main loop

The main loop code is the following:

```
pro MainLoop, isfd
    connections = IdlFindSockets(isfd)
    while (1 eq 1) do begin
        fd = IdlSelect(connections)
        if (fd ne -1) then begin
            if (fd eq isfd) then begin
                IsdatEvent
            endif
        endif
    endwhile
end
```

where we first find all socket connections and then enter into an infinite loop. In the loop, we select one socket, and if it is OK, then we wait for an ISDAT "event" for ever.

An ISDAT "event" usually consist of an "update" from the *time manager* (see [Ref. 4]). The *IsdatEvent* function is the following:

```
pro IsdatEvent
    common info, info

    info = GetTmInfo()
    if (info.new eq 1) then update
end
```

where we first obtain information about the current time interval and if the time interval has been changed, we call the *update* function to modify our display. The *update* code is described in section 2.2.3.

2.2.3 Updating the display

When *update* is called we have to request new data from the ISDAT server by:

```
    desc = GetTaggedData(db, info.start, info.interval, dspec, 0L)
```

where db is the data base descriptor, info.start is the start time, info.interval is the time interval, dspec describes the data we want, and 0L tells the server to return as many samples as needed. See also Appendix A for a description of the library functions. The data is returned in a quite general and complex structure *desc*. This structure is described in [Ref. 8].

Then we define a fixed title to be printed on the plot:

```
    title = 'CSDS_UI IDL test plot'
```

If the ISDAT server reports an error in the *desc* structure, we simply print out the title and the error message and then report to the time manager that we are ready by:

```
    if (desc.error ne 0) then begin
        erase
        title = title + '!C' + IdlErrorString(desc.error)
        xyouts, 0.2, 0.5, title, charsize = 2.0, /normal
    endelse
    IdlDone
```

If the ISDAT server returns no error message, we print warnings if any, label axes, plots the data (desc.data) versus time (desc.time), and report back that we are ready IdlDone). The whole *update* functions is:

```
pro update
    common info, info
    common spec, dspec
    common database, db
    xrange = dblarr(2)

    desc = GetTaggedData(db, info.start, info.interval, dspec, 0L)

    title = 'CSDS_UI IDL test plot'

    if (desc.error ne 0) then begin
        erase
        title = title + '!C' + IdlErrorString(desc.error)
        xyouts, 0.2, 0.5, title, charsize = 2.0, /normal
    endif else begin ; If no errors encountered, plot(/or treat) the data.
        if (desc.warning ne 0) then begin
            print, 'warning: ', IdlWarningString(desc.warning)
        endif

        xtitle = 'seconds'
        ytitle = desc.quantity(0) + '  (' + desc.units(0) + ')'
        xrange(0) = 0.0
        xrange(1) = info.interval
        wset, 1
        plot, desc.time, desc.data, $
            title = title, xtitle = xtitle, ytitle = ytitle, $
            xrange = xrange, charsize = 2.0
        wshow, 1, 1
    endelse
    IdlDone
end
```

3 Executing the ISDAT IDL client

For the execution of the IDL client it is recommended to use a shell script:

```
#!/bin/sh
```

```
IMSLIDL_STARTUP=loadlib.pro
export IMSLIDL_STARTUP
IDL_STARTUP=$IMSLIDL_STARTUP
export IDL_STARTUP

# find out OS major version
if [ -f /bin/truss ]; then MAJOR=5; else MAJOR=4; fi

if [ "$OSMAJOR" = 5 ]; then
    idl bat_cuit.pro
else
    imslidl bat_cuit.pro
fi
```

The shell script should be given the preferred name of your client, in our case we name it *cuit*. The purpose of the *cuit* shell script is to:

- load the ISDAT/IDL library
- Identify the operative system version used on the local platform
- Start the client via the file bat_cuit.pro

In our case the bat_cuit.pro file consist of one line:

```
    .run cuit
```

Thus to start the *cuit* client do:

```
    >cuit
```

from a terminal window. If properly done, a *cuit* window should appear, and the plotted data should be updated as the time manager *cuitm* changes the time interval.

To stop the *cuit* client do

```
>ctrl-c
```

from the terminal window, and then

```
>exit
```

from IDL.

4 Advanced ISDAT IDL clients

Using and expanding the *cuit* example described in section 2 you can make use of most of the possibilities offered by IDL, including graphics, signal processing, building of user interfaces etc. For more information on IDL, please consult your IDL manuals. The source code for a slightly more complex ISDAT/IDL client, the source code for the IDL client *p3* is included in Appendix C. The *p3* client has a simple user interface that allows the user to select the parameters to be plotted. It is capable of plotting up to three panels in one window. The *p3* client is also capable to dynamically build parameter menues using information supplied by the ISDAT server.

5 Known bugs

There are currently no known bugs.

A ISDAT/IDL Functions

```
This directory contains the interface between ISDAT and IMSLIDL.
It has been tested on SunOS 4.1.3 and SunOS 5.3.


The islib.pro defines the following functions:

s = GetTmInfo()
    return - structure {new, start, interval,
                        project, member, instrument,
                        contend}

s = GetTaggedData(db, start, interval, spec, samples)
    db - long, database handle
    start - double, start time
    interval - double, time interval
    spec - int array with 7 elements, data specification
    samples - long, max number of samples to return, 0 gives all within interval
    return - structure {error, warning, data, tbase, time, rate,
                        minval, maxval, quantity, units, scale}

The libIdl.so.1.0 library is loaded by load.pro and contains
the following functions:

----- Is functions (IsFunc.c)

fd = IdlInitialize()
    return - int with the file descriptor for the time manager connection

fdarr = IdlFindSockets(firstfd)
    firstfd - int, start search at this file descriptor
    return - int array with the file descriptors for all socket connections

err = IdlSelect(fdarr)
    fdarr - int array with file descriptors to select on, see select(2)
    return - int with ready file descriptor, -1 on signal EINTR

----- Is time manager info functions (IsFunc.c)

new = IdlGetTmInfo()
    return - int, 1 if time manager update, 0 otherwise

start = IdlTmInfoStart()
    return - double, time manager start time

interval = IdlTmInfoInterval()
    return - double, time manager time interval

end = IdlTmInfoContEnd()
    return - double

project = IdlTmInfoProject()
    return - int, time manager project
```

```
member = IdlTmInfoMember()
    return - int, time manager member

member = IdlTmInfoInstrument()
    return - int, time manager instrument

mode = IdlTmInfoMode()
    return - int

----- Is procedures (IsFunc.c)

IdlDone

----- Db functions (DbFunc.c)

db = IdlOpen([database])
    database - string, database name, no argument uses DATABASE
       environment variable
    return - long, database handle

----- Db data functions (DbData.c)

err = IdlGetTaggedData(db, start, interval, spec, samples)
    db - long, database handle
    start - double, start time
    interval - double, time interval
    spec - int array with 7 elements, data specification
    samples - long, max number of samples to return, 0 gives all within interval
    return - int, error code, 0 if no error occured


warning = IdlDataWarning()
    return - long

data = IdlDataValue()
    return - float array, dimension and size depends on the data

tstart = IdlDataStart()
    return - double, time of first sample

tag = IdlDataTimeTag()
    return - double array, first tag is always zero

units = IdlDataUnits()
    return - string array

rate = IdlDataRate()
    return - double, sampling rate

quantity = IdlDataQuantity()
    return - string array

min = IdlScaleMin()
    return - float array

max = IdlScaleMax()
```

```
    return - float array

scale = IdlDataScaleType()
    return - string array

----- Db conversion functions (DbConv.c)

name = IdlSpec2Name(db, spec)
    db - long, database handle
    spec - int array with 7 elements, data specification
    return - string array

spec = IdlName2Spec(db, name)
    db - long, database handle
    name - string array
    return - int array with 7 elements, data specification

str = IdlErrorString(err)
    err - int, error code
    return - string

str = IdlWarningString(warning)
    warning - long, warning bit mask
    return - string

----- Db query functions (DbQuery.c)

strarr = IdlQueryMenu(db, project, member, instrument, title)
    db - long, database handle
    project - int
    member - int
    instrument - int
    title - string
    return - string array to be passed to XPDMENU

spec = IdlMenuValue2Spec(valstr)
    valstr - string
    return - int array with 7 elements, data specification
```

B cuit.pro source code

```
pro update
    common info, info
    common spec, dspec
    common database, db
    xrange = dblarr(2)

    desc = GetTaggedData(db, info.start, info.interval, dspec, 0L)

    title = 'CSDS_UI IDL test plot'

    if (desc.error ne 0) then begin
        erase
        title = title + '!C' + IdlErrorString(desc.error)
        xyouts, 0.2, 0.5, title, charsize = 2.0, /normal
    endif else begin  ; If no errors encountered, plot(/or treat) the data.
        if (desc.warning ne 0) then begin
    print, 'warning: ', IdlWarningString(desc.warning)
        endif

xtitle = 'seconds'
ytitle = desc.quantity(0) + '  (' + desc.units(0) + ')'
xrange(0) = 0.0
        xrange(1) = info.interval
wset, 1
        plot, desc.time, desc.data, $
      title = title, xtitle = xtitle, ytitle = ytitle, $
      xrange = xrange, charsize = 2.0
        wshow, 1, 1
    endelse
    IdlDone
end


pro IsdatEvent
    common info, info

    info = GetTmInfo()
    if (info.new eq 1) then update
end

pro MainLoop, isfd
    connections = IdlFindSockets(isfd)
    while (1 eq 1) do begin
        fd = IdlSelect(connections)
        if (fd ne -1) then begin
    if (fd eq isfd) then begin
        IsdatEvent
    endif
        endif
    endwhile
end
```

```
; main program

    common info, info
    common spec, dspec
    common database, db

    tmInfoFd = IdlInitialize() ; must be executed first in program
    !y.margin = [3.0, 3.0]
    window, 1

    ; Open connection to data base handler.
    db = IdlOpen()
    if (db eq 0) then begin
        print, 'no database handler running'
        exit
    endif

    info = GetTmInfo()

    name = ['CSDS_PP', 'C1', 'EFW', 'E_dusk', '', '', '']
    print, 'name=', name
    dspec = IdlName2Spec(db, name)

    MainLoop, tmInfoFd
end
```

C p3.pro source code

```
function ExtractSpec, spec, panel
    sp = intarr(7)

    for i = 0, 6 do sp(i) = spec(panel, i)
    return, sp
end

pro Update, panel
    common info, info
    common specification, spec, valid, current
    common widgets, w
    common database, db
    xrange = dblarr(2)

    if (valid(panel) eq 0) then return
    spec(panel, 0) = info.project
    spec(panel, 1) = info.member
    spec(panel, 2) = info.instrument
    sp = ExtractSpec(spec, panel)
    version = info.dataversion
    desc = GetTaggedData(db, info.start, info.interval, sp, 0L, version)

    wset, w.win(panel)
    title = ''
    specname = IdlSpec2Name(db, sp);
    for i = 0, 6 do title = title + specname(i) + ' '

    if (desc.error ne 0) then begin
erase
title = title + '!C' + IdlErrorString(desc.error)
xyouts, 0.2, 0.5, title, charsize = 2.0, /normal
    endif else begin
; If no errors encountered, plot(/or treat) the data.
if (desc.warning ne 0) then begin
    print, 'warning: ', IdlWarningString(desc.warning)
endif

xtitle = 'seconds'
ytitle = desc.quantity(0) + '  (' + desc.units(0) + ')'
xrange(0) = 0.0
xrange(1) = info.interval
dataSize = size(desc.data)
if (desc.dimension eq 0) and (desc.rank eq 0) then begin
    plot, desc.time, desc.data, $
  title = title, xtitle = xtitle, ytitle = ytitle, $
  xrange = xrange, charsize = 1.5, color = 1
endif else if (desc.dimension eq 0) and (desc.rank eq 1) then begin
    yrange = [min(desc.data), max(desc.data)]
    plot, desc.time, desc.data(*, 0), $
  title = title, xtitle = xtitle, ytitle = ytitle, $
  xrange = xrange, yrange = yrange, $
  charsize = 1.5, color = 1
    oplot, desc.time, desc.data(*, 1), color = 2
    oplot, desc.time, desc.data(*, 2), color = 3
endif else begin
    s2 = dataSize(2)
    ydata = findgen(s2)
    m = desc.minval(0)
    k = (desc.maxval(0) - desc.minval(0)) / float(s2)
    for i = 0, s2 - 1 do ydata(i) = k * i + m
    surface, desc.data, desc.time, ydata, $
     title = title, xtitle = xtitle, ytitle = ytitle, $
     charsize = 2.0
endelse
    endelse
    IdlDone
    ; make sure that the X-connection gets flushed
    empty
end


pro IsdatEvent
    common info, info

    info = GetTmInfo()
    if (info.new eq 1) then begin
        for panel = 0, 2 do Update, panel
    endif
end

pro WidgetEvent, ev
    common info, info
    common specification, spec, valid, current
    common widgets, w
    common database, db

    type = tag_names(ev, /structure)
    if (type eq 'WIDGET_BUTTON') then begin
for panel = 0, 2 do begin
    if (ev.id eq w.b(panel)) then begin
current = panel
for lab = 0, 2 do widget_control, w.l(lab), sensitive = 0
widget_control, w.l(current), sensitive = 1
return
    endif
endfor
widget_control, ev.id, get_uvalue = value
vsize = size(value)
vdim = vsize(0)
vtype = vsize(vdim + 1)
if (vtype ne 0) then begin
    if (strmid(value, 0, 5) eq 'spec.') then begin
widget_control, w.v(current), set_uvalue = value
spec(current, *) = IdlMenuValue2Spec(value)
spec(current, 0) = info.project
spec(current, 1) = info.member
spec(current, 2) = info.instrument
sp = ExtractSpec(spec, current)
name = IdlSpec2Name(db, sp);
```

```
str = ''
for i = 0, 6 do str = str + name(i) + ' '
widget_control, w.v(current), set_value = str
valid(current) = 1;
Update, current
    endif
endif
    endif
end

pro MainLoop, isfd
    common widgets, w

    connections = IdlFindSockets(isfd)
    while (1 eq 1) do begin
fd = IdlSelect(connections)
if (fd ne -1) then begin
    if (fd eq isfd) then begin
IsdatEvent
    endif else begin
ev = widget_event(w.base, /nowait, bad_id = bad)
if (bad ne 0) then exit
if (ev.id ne 0) then WidgetEvent, ev
    endelse
    ; make sure the imslidl X connection gets flushed
    ev = widget_event(w.base, /nowait, bad_id = bad)
    if (bad ne 0) then exit
    if (ev.id ne 0) then WidgetEvent, ev
endif
    endwhile
end

; main program

    common info, info
    common specification, spec, valid, current
    common database, db
    common widgets, w

    tmInfoFd = IdlInitialize() ; must be executed first in program

    ; setup color table, 0 black, 1 white, 2 red, 3 green, 4 blue, 5 yellow
    red = [0, 1, 1, 0, 0, 1]
    green = [0, 1, 0, 1, 0, 1]
    blue = [0, 1, 0, 0, 1, 0]
    tvlct, 255 * red, 255 * green, 255 * blue

    ; Define some variables.
    current = 0
    valid = intarr(3)
    spec = intarr(3, 7)
    w = {base: 0L, controlbase: 0L, $
 b: [0L, 0L, 0L], l: [0L, 0L, 0L], v: [0L, 0L, 0L], $
 draw: [0L, 0L, 0L], $
 win: [0, 0, 0]}
```

```
    !y.margin = [2.0, 1.5]

    ; Open connection to data base handler.
    db = IdlOpen()
    if (db eq 0) then begin
print, 'no database handler running'
exit
    endif

    info = GetTmInfo()
    menuarr = IdlQueryMenu(db, $
   info.project, info.member, info.instrument, $
   'select')
    w.base = widget_base(title = 'p3', column = 1, space = 0)
    xpdmenu, menuarr, w.base
    w.controlbase = widget_base(w.base, row = 1, frame = 1)
    w.draw(0) = widget_draw(w.base, xsize = 600, ysize = 230)
    w.draw(1) = widget_draw(w.base, xsize = 600, ysize = 230)
    w.draw(2) = widget_draw(w.base, xsize = 600, ysize = 230)

    currbase = widget_base(w.controlbase, column = 1, frame = 1, space = 0)
    wid = widget_label(currbase, value = 'current channel:')
    currsub = widget_base(currbase, row = 1, exclusive = 1, space = 0)
    w.b(0) = widget_button(currsub, value = ' 0 ')
    w.b(1) = widget_button(currsub, value = ' 1 ')
    w.b(2) = widget_button(currsub, value = ' 2 ')

    qtybase = widget_base(w.controlbase, column = 1, frame = 1, space = 0)
    wid = widget_label(qtybase, value = 'quantities')
;    qtysub = widget_base(qtybase, column = 2, space = 0) ; imslidl
    qtysub = widget_base(qtybase, row = 3, space = 0) ; idl
    w.l(0) = widget_label(qtysub, value = 'channel 0:')
    w.v(0) = widget_label(qtysub, value = '                              ')
    w.l(1) = widget_label(qtysub, value = 'channel 1:')
    w.v(1) = widget_label(qtysub, value = '                              ')
    w.l(2) = widget_label(qtysub, value = 'channel 2:')
    w.v(2) = widget_label(qtysub, value = '                              ')
    widget_control, /realize, w.base
    widget_control, w.b(0), set_button = 1
    widget_control, w.l(1), sensitive = 0
    widget_control, w.l(2), sensitive = 0
    for panel = 0, 2 do begin
widget_control, get_value = win, w.draw(panel)
w.win(panel) = win
    endfor

    MainLoop, tmInfoFd
end
```