Advanced Multi-Mission Operations System Instrument Toolkit

An open source instrument and small satellite operations toolkit

Multimission Ground System and Services Office – Instrument Data Systems
Presented by Michael Joyce[1]

[1]Jet Propulsion Laboratory, California Institute of Technology

AIT Overview

AIT is an instrument I&T and operations toolkit supporting uplink and downlink

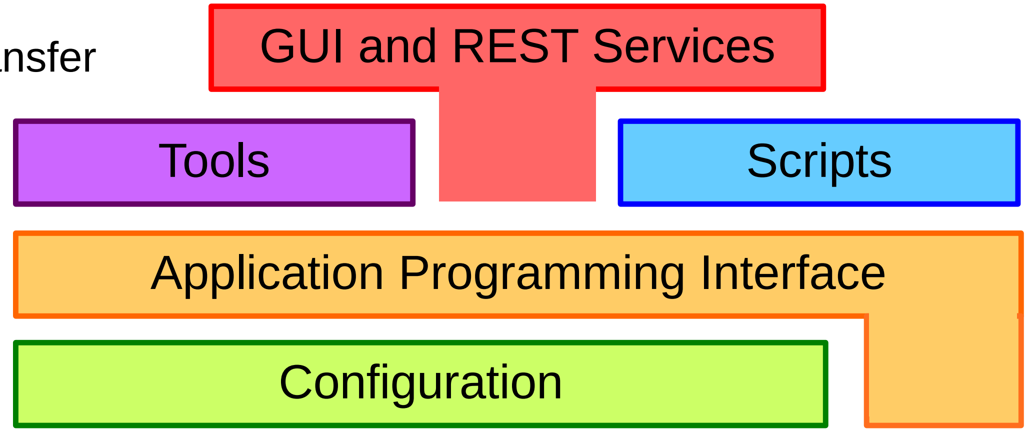
- Python-based GDS/MOS/EGSE, command, telemetry, and sequencing toolkit
- Initially developed for Vehicle Cabin Atmosphere Monitor (VCAM) and Electronic Nose (ENose) ISS instruments circa 2008.
- Updated and expanded for OCO-3 (TBD launch) and now ECOSTRESS (2018 launch). Baseline for ISS S.A.M instrument (a VCAM follow-on), Cold Atom Lab (CAL), M2020 MOXIE EGSE, MAIA IOS, and Lunar IceCube.
- Extended to support CCSDS SLE and CFDP specifications for Small Satellites in 2017/18

AIT Feature Highlights

- Applicable in both EGSE (I&T) and Operations settings
- Modern web-based interface (HTML5/CSS/JS)
- Python scripting
- Efficient packet evaluation
- Standard file formats whenever possible (e.g. .pcap, .yaml)
- YAML Command and Telemetry Dictionaries
    - Used by AIT GDS tools for:
        - CMD/TLM Processing and Validation
        - Sequencing
        - Command Builder UI
        - Telemetry Display
        - Monitoring
        - Plotting
        - etc.

AIT Component Architecture

- Layered component architecture
- Starts with a easily human readable, *declarative* configuration (what, not how)
- Unix philosophy: Simple file formats (SQL/NoSQL, elastic DBs not precluded); small components and APIs that do one thing well
- Centralized, syslog and pcap based logging (built in AIT API)
- TCP, UDP/IP for real-time data transfer
- Web-based Client / Server
  - HTTP Web Servers
  - REST APIs



AIT Commanding

Command Dictionary

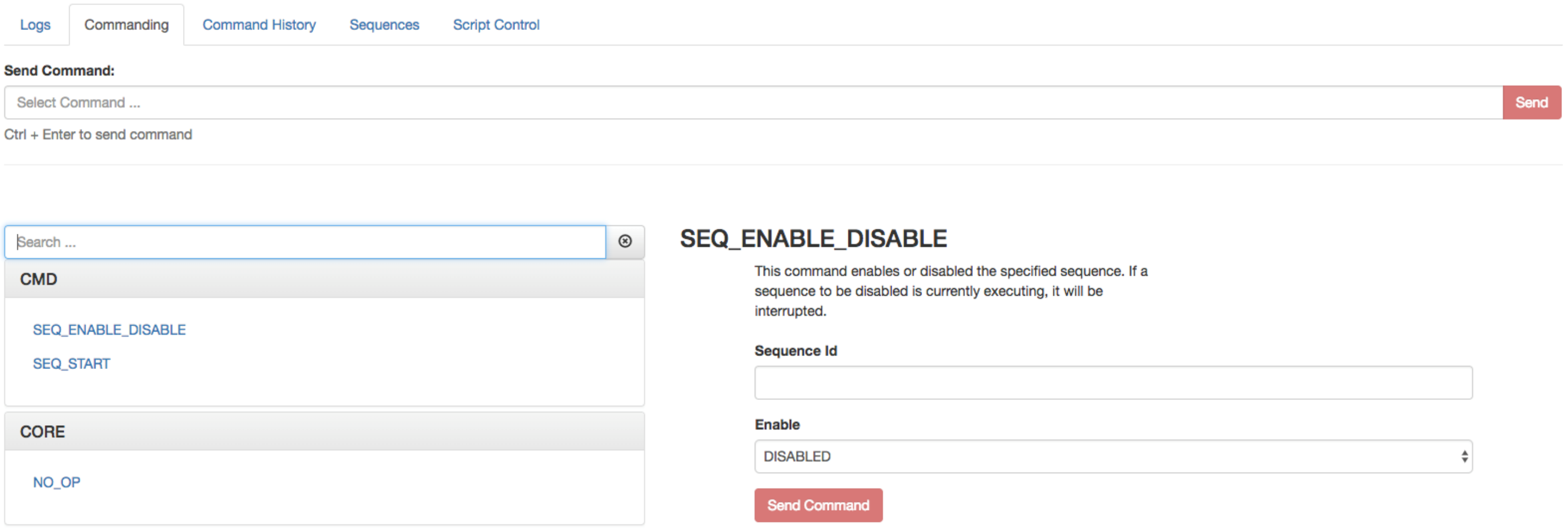
- Schema documents and represents commands with enough detail to verify units, ranges, etc. and encode to or decode from binary
- Simple plain-text representation of schema in YAML (YAML Ain't Markup Language)
    - Easy to view / edit with any text editor and many other tools
    - Easy to compare and difference
    - Track changes and evolution over time in SCM system
- Single representation feeds many different tools
    - Encode to / decode from binary format
    - Validate syntax and parameters
    - Monitoring GUI functionality
    - Generate FSW source code
    - LaTeX to PDF documentation

```yaml
--- !Command
name:      CORE_SET_OP_MODE
opcode:    0x1234
subsystem: CORE
desc:      |
  This command sets the operational mode.

arguments:
  - !Argument
    name:  mode
    desc:  Mode
    units: none
    type:  U8
    bytes: 0
    enum:
      0: SAFE
      1: IDLE
      2: SCANNING
      3: SCIENCE
```

config/cmd.yaml

AIT Commanding

Monitoring GUI



Send Command:

Select Command ... [Send]

Ctrl + Enter to send command

SEQ_ENABLE_DISABLE

This command enables or disabled the specified sequence. If a sequence to be disabled is currently executing, it will be interrupted.

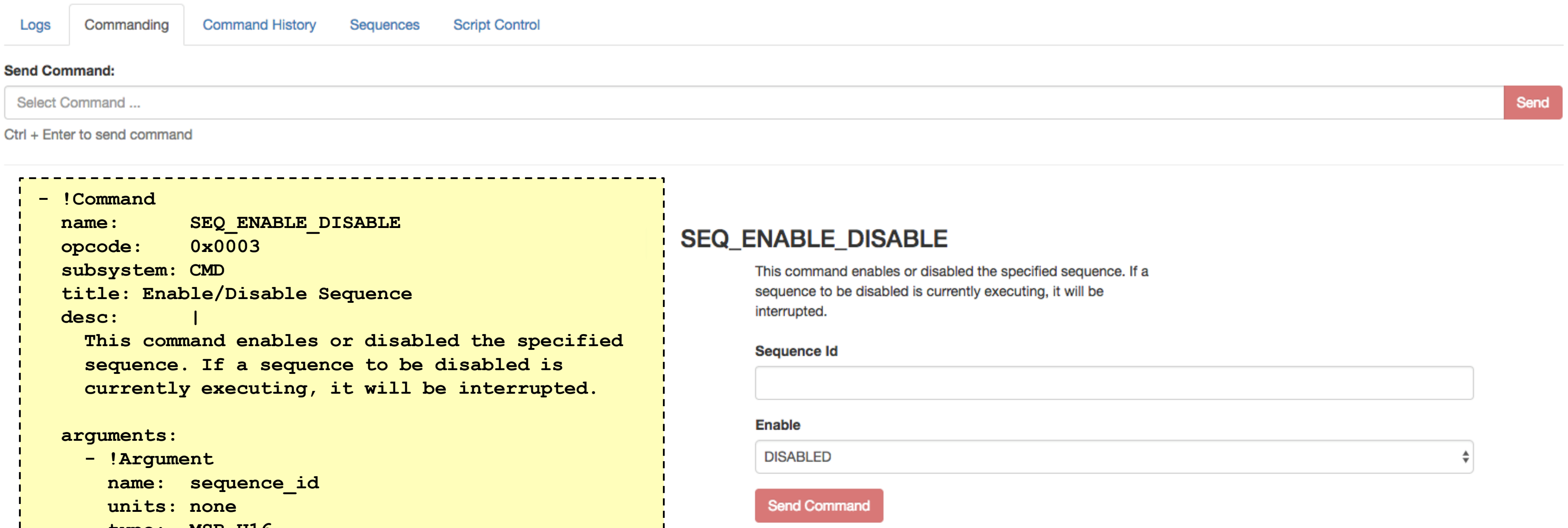
Sequence Id

Enable

DISABLED

AIT Commanding

Monitoring GUI



SEQ_ENABLE_DISABLE

This command enables or disabled the specified sequence. If a sequence to be disabled is currently executing, it will be interrupted.

Sequence Id:

Enable: DISABLED

```
- !Command
  name:      SEQ_ENABLE_DISABLE
  opcode:    0x0003
  subsystem: CMD
  title: Enable/Disable Sequence
  desc:      |
    This command enables or disabled the specified
    sequence. If a sequence to be disabled is
    currently executing, it will be interrupted.

  arguments:
    - !Argument
      name:  sequence_id
      units: none
      type:  MSB_U16
      bytes: [0,1]

    - !Argument
      name:  enable
      units: none
      type:  U8
      bytes: 2
      enum:
        0: DISABLED
        1: ENABLED
```

AIT Sequencing

Relative-Time Sequences

- Consists of time offset and command to execute
- Can be executed from ground via Monitoring UI or API
- Can be encoded for upload to FSW

```
2 CORE_SET_OP_MODE INSTRUMENT_SAFE
2 CORE_SET_OP_MODE SCIENCE

# boot V5 interface
2  CORE_BOOT_V5
15 CORE_NO_OP

# set GPS interface options
5 GPS_SET_PROGRAM_INTERFACE PVT_SOLUTION 0
5 GPS_SET_PROGRAM_INTERFACE GPS_TIME 0
5 GPS_SET_PROGRAM_INTERFACE SGR_STATUS 0
```

AIT Telemetry

Telemetry Dictionary

CCSDS Primary Header						
	Packet Identification			Packet Sequence Control		
Packet Version Number	Type	Secondary Header Flag	ApID	Sequence Flags	Sequence Count	Packet Length
3 bits	1 bit	1 bit	11 bits	2 bits	14 bits	16 bits
2 bytes				2 bytes		2 bytes

```
- !Packet
  name: CCSDS_Packet
  type: ethernet
  desc: FSW 1553 Telemetry
  headers:
    - !Header
      name: CCSDS_Primary
      fields:
        - !Field
          name:       version
          desc:       Indicates CCSDS Version-1 (does not change)
          bytes:       0
          type:       U8
          mask:       0xE0

        - !Field
          name:       type
          desc:       |
            Distinguishes between core and payload packet types
            to extend the APID space to 4032
          bytes:       0
          type:       U8
          mask:       0x10
          enum:
            0: 'Core'
            1: 'Payload'

        - !Field
          name:       secondary_header_flag
          desc:       |
            Indicates whether, or not, a Secondary Header follows
            the primary header (always set to 1)
          bytes:       0
          type:       U8
          mask:       0x08
          enum:
            0: 'Not Present'
            1: 'Present'

        - !Field
          name:       apid
          desc:       |
            Used in conjunction with Type to define the Logical
            Data Path
          bytes:       [0, 1]
          type:       MSB_U16
          mask:       0x07FF

        - !Field
          name:       sequence_flags
          desc:       |
            When sending commands, the sequence flags must be
            marked as unsegmented data. All other PL packets may
            be per source/destination ICDs.
          bytes:       2
          type:       U8
          mask:       0xC0
          enum:
            0: 'Continuation Segment'
            1: 'First Segment'
            2: 'Last Segment'
            3: 'Unsegmented'
```

config/tlm.yaml

AIT Telemetry

Telemetry Dictionary Features

Telemetry dictionary also provides ability to specify:

- Constants
- Functions
- History
- DN to EU
- Muxed Data
- Masks
- Imports

```
- !Packet
  name:   AMR_St
  desc:   |
    See AMR Com

  constants:
    A:    371.81
    B:   -4.850e
    C:    1.086e
    D:   -1.239e
    RL:  1000.0
    RH:  5000.0

  functions:
    R(dn): RL +
    T(dn): A +

  history:
    - VX0
    - VX1
    - VX2
    - VX3
    - VX4

  !include 1553_ehs.yaml
```

```
- !Field
       name:    VX6
       desc:    VFC (+12V)
       dntoeu:
         equation: 11.53 * raw.VX6 / history.VX0
         units:    volts
         when:     history.VX0 > 2000
       type:    MSB_U16
       when:    HKMux1 == 0x6e
```

config/tlm.yaml

AIT Telemetry

Telemetry Dictionary Features

Telemetry dictionary also provides ability to specify:

- Constants
- Functions
- History
- DN to EU
- Muxed Data
- Masks
- Imports

```
- !Packet
  name:  AMR_St
  desc:  |
    See AMR Com

  constants:
    A:   371.81
    B:  -4.850e
    C:   1.086e
    D:  -1.239e
    RL: 1000.0
    RH: 5000.0

  functions:
    R(dn): RL +
    T(dn): A +

  history:
    - VX0
    - VX1
    - VX2
    - VX3
    - VX4

  !include 1553_ehs.yaml
```

```
- !Field
      name:    RT2
      bytes:   '@prev'
      desc:    RT2 External Temp
      dntoeu:
        equation: T(raw.RT2)
        units:    Kelvin
        when:     (history.RT1 - history.RT0) > 3000
      type:    MSB_U16
      mask:    0xFF00
      when:    HKMux1 == 18
```

config/tlm.yaml

AIT Telemetry

Limits

- Monitor telemetry streams for invalid values

- Define “warning” and “error value ranges
- Specify mnemonics that are out of limit

- Notify users in Monitoring UI of invalid values
- Automatically send text and email alerts when limit trips are detected.

```
- !Limit
  source:  1553_EHS.AnalogsCurr_STAR_TRACKER
  desc:    Overcurrent - Star Tracker
  units:   amperes
  upper:
    warn:  0.43
    error: 0.52

- !Limit
  source:  1553_EHS.AnalogsVoltage_V_GA
  desc:    Voltage Bank A
  units:   volts
  lower:
    error: 23.0
    warn:  27.0
  upper:
    warn:  29.0
    error: 35.0

- !Limit
  source:  1553_EHS.SRUSWBootFailure
  desc:    SRU S/W Boot Failure
  value:
    error: FAILED
```

config/limits.yaml

Configurable Telemetry Displays

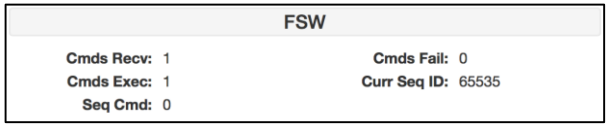
It's just HTML

gui/telem.html

```
<h4 class="telem-group-title">FSW</h4>
  <table class="telem col2">
    <tr> <td>Cmds Recv:   <td><ait-field name="CmdsRcvd">
         <td>Cmds Fail:   <td><ait-field name="CmdsFailed">
    <tr> <td>Cmds Exec:   <td><ait-field name="CmdsExec">
         <td>Curr Seq ID: <td><ait-field name="CurrSeqID">
    <tr> <td>Seq Cmd:     <td><ait-field name="SeqCmdOffset">
  </table>
</div>
```

config/tlm.yaml

```
  - !Field                        - !Field
      name: CmdsRcvd                 name: CurrSeqID
     bytes: [297,298]               bytes: [303,304]
      type: MSB_U16                   type: MSB_U16

  - !Field                        - !Field
      name: CmdsFailed               name: SeqCmdOffset
     bytes: [299,300]               bytes: [305,306]
      type: MSB_U16                   type: MSB_U16

  - !Field
      name: CmdsExec
     bytes: [301,302]
      type: MSB_U16
```

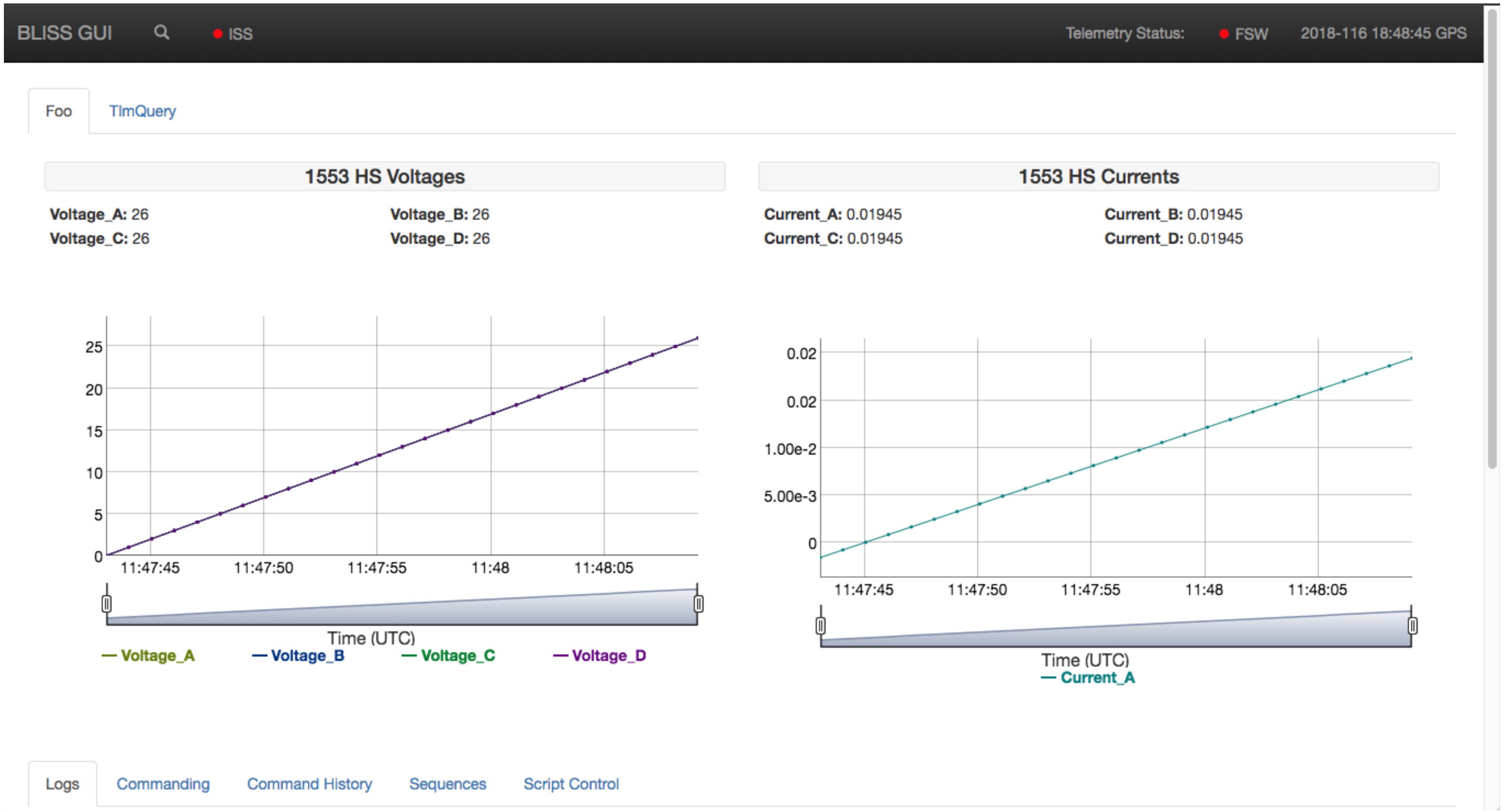


FSW

Cmds Recv:	1	Cmds Fail:	0
Cmds Exec:	1	Curr Seq ID:	65535
Seq Cmd:	0		

AIT Telemetry

Monitoring Display



1553 HS Voltages

Voltage_A: 26
Voltage_B: 26
Voltage_C: 26
Voltage_D: 26

1553 HS Currents

Current_A: 0.01945
Current_B: 0.01945
Current_C: 0.01945
Current_D: 0.01945

AIT API and Payload Scripting

Low-Level Script Interface: It’s Python!

```
from ait.core     import log
from ait.core.api import APITimeoutError, CmdExecError, wait
import myInstrument

try:
    p = myInstrument.api.Payload()
    p.OVERRIDE_NEXT_CMD()
    wait("p.ehs.CmdOverrideEnabled == 'OVERRIDE_ENABLED'", timeout=15)

    p.execute('RELEASE_LOCK SWITCH_ON PRIMARY')
    wait("p.ehs.PMALockState == 'UNLOCKED'", timeout=15)

    p.execute('SET_RELEASE_THRESHOLD', 0xffff)
    wait("p.ehs.PMAReleaseThreshold == '0xffff'", timeout=15)

    p.execute('SWITCH_POWER', 'SWITCH_ON')
    wait("p.ehs.PCEPMAOn", timeout=15)

    log.info("SUCCESS: Lock Deploy Executed")

except (APITimeoutError, CmdExecError) as e:
    log.error(e.msg)
```

Scratching the Surface

- Command and Data Handling Tables
  - Configurable FSW data structures
- Event Verification Records
  - Decode FSW EVR codes into human-readable records
- AIT – Goddard’s Core Flight Software bridge
  - Building blocks for communication with open source FSW
- DSN SLE and File Interfaces
  - Return All Frames, Return Channel Frames, and Forward CLTU
  - CFDP Class 1

AIT Documentation, Support, Installation

- Visit
  - https://github.com/NASA-AMMOS
- Mailing Lists
  - ait-dev@googlegroups.com
  - https://groups.google.com/forum/#!forum/ait-dev
- Documentation
  - https://ait-core.readthedocs.io/en/latest/
  - https://ait-gui.readthedocs.io/en/latest/
- Install

```
$ pip install ait-core
$ pip install ait-gui
```

AIT Command and Data Handling Tables

Onboard Target Table

Target	Latitude	Longitude	Elevation
1	34.200° N	118.18° W	0.39 km
…	…	…	…
1335	34.406° S	150.879° E	0.03 km

AIT
Encode / Decode

Binary Data

```
typedef struct {
        ushort16 targetID;
        double64 latDegs;
        double64 lonDegs;
        double64 altitudeMeters;
        uchar8   enabled;
    } target_t;
…
```

config/table.yaml

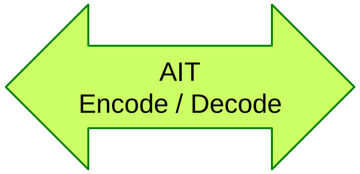
```
--- !FSWTable
name: targets
delimiter: ","
columns:
  - !FSWColumn
    name:   TARGET_ID
    desc:   Targets target ID
    format: "%u"
    units: none
    type:   MSB_U16

  - !FSWColumn
    name:   LAT_DEGS
    desc:   Targets latitude degrees
    format: "%20.6f"
    units: deg
    type:   MSB_D64
- !FSWColumn
    name:   LON_DEGS
    desc:   Targets longitude degrees
    format: "%20.6f"
    units: deg
    type:   MSB_D64

  - !FSWColumn
    name:   ALTITUDE_METERS
    desc:   Targets altitude meters
    format: "%20.6f"
    units: meters
    type:   MSB_D64
```

AIT Command and Data Handling Tables

Example Tables and Products

Onboard Target Table

Target	Latitude	Longitude	Elevation
1	34.200° N	118.18° W	0.39 km
…	…	…	…
1335	34.406° S	150.879° E	0.03 km



Binary Data

```
typedef struct {
        ushort16 targetID;
        double64 latDegs;
        double64 lonDegs;
        double64 altitudeMeters;
        uchar8   enabled;
    } target_t;
…
```

Configuration-based approach allows projects to use the same AIT tools for multiple C&DH onboard data tables:

1. CRC
2. Downlink Windows
3. Fault Response
4. Keep Out Zones
5. Line-of-Sight Errors
6. Memory Scrub
7. Schedule
8. Targets
9. Error Logs