



IceStorm Series II Enclosure


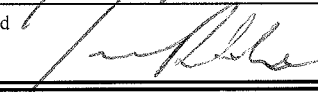
Software Interface Design Description

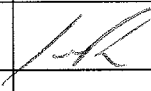

CI No. IDD07753-01

EOS Confidential Proprietary Document

EOS SPACE SYSTEMS PTY LIMITED
111 CANBERRA AVENUE GRIFFTH, ACT, 2603
A.B.N. 25 100 248 253
TELEPHONE 61 2 6222 7900
FACSIMILE 61 2 6299 7687

DOCUMENT CONTROL

Prepared	Signed Michael Forman	Date 26/10/06
Checked	Signed 	Date 26/10/06
Approved	Signed 	Date 26/10/06

1	26/10	Original Issue		
Issue	Date	Description	Checked	Approved

Index	
-------	--

TABLE OF CONTENTS

1. SCOPE	5
1.1 Identification	5
1.2 Background	5
2. OVERVIEW	6
3. SAFETY SYSTEM	7
3.1 Dome States	7
3.2 Basic States	9
3.3 Lifeguard / Watchdog	10
3.4 Personnel Safe State	13
3.5 E-Secure Inputs	14
4. SCHEMA	15
5. FIELD DEFINITIONS	27
5.1 Server	29
5.1.1 Commands	29
5.1.2 Attributes	33
5.2 Common/Shared among Multiple Devices	34
5.2.1 Attributes	34
5.3 Azimuth	37
5.3.1 Commands	37
5.3.2 Attributes	38
5.4 Shutter	44
5.4.1 Individual Shutter	44
5.4.1.1 Commands	44
5.4.1.2 Attributes	50
5.4.2 Shutters	57
5.4.2.1 Commands	57
5.4.2.2 Attributes	59

5.5	Vent Doors	61
5.5.1	Commands	61
5.5.2	Attributes	62
5.6	Safety	66
5.6.1	Commands	66
5.6.2	Attributes	69
5.7	Cooling	74
5.7.1	Commands	74
5.7.2	Attributes	75
5.8	Lighting Circuit	78
5.8.1	Commands	78
5.8.2	Attributes	78
5.9	Recirculation Fans and the Snow Melt Cable, (Other Digital I/O Channels)	80
5.9.1	Commands	80
5.9.2	Attributes	80

1. SCOPE

1.1 Identification

This document describes the messaging interface for the IceStorm Series II Enclosure. This interface lies between the server software that controls the enclosure and higher-level software both within the system and external to the system.

This document refers to version 0.10 of the Schema.

1.2 Background

The IceStorm Series II Enclosure is controlled and monitored by the *IceStorm Server*, a software component that allows any number of client applications, either remote or local, to connect to the server. Once connected to the server, clients may issue *commands* to the server or retrieve specified data items, or *attributes*, from the server. Commands on the server may be completed either synchronously or asynchronously. Attributes may also be delivered either synchronously or asynchronously.

The complete set of commands and attributes relating to the IceStorm Enclosure is expressed in an ASCII document, the *DML schema*. The schema is written in Device Meta Language (DML), a special-purpose syntax designed for the purpose of writing schemas.

2. OVERVIEW

At its core the IceStorm Series II dome is controlled by multiple, semi-independent microcontroller based processing units (nodes). These nodes are interconnected via a CANopen¹ communications bus and under normal operating conditions, receive commands and transmit status data from/to the main control computer along this bus. This distributed processing architecture allows the main control computer to issue commands and monitor the results, but alleviates the need for its involvement in the actual low level control of the electro-mechanical systems.

There are two CAN buses in use in the dome; one for the main electro-mechanical components, the other for the air-conditioning system.

Electro-mechanical CAN bus:

- Azimuth controller
- Front and Rear shutter controllers. Depending on the configuration the shutters Motor Drives may also appear on this bus
- 4* Vent Door controllers
- Safety node
- Digital I/O (connected to devices such as house lights, fans, snow melt)

Air-conditioning CAN bus:

- 3 * Fan Control Units (FCU), each consisting of
 - Coil temperature sensor
 - Cooling/heating output control
 - Loop controller, controls cooling valve and fan
- Nominally 3* Temperature, Humidity and Dewpoint (THüD) sensors. Note that there may be any number of these THüDs installed in various locations as required.

The IceStormServer acts as a proxy to these devices, providing a simpler interface and hiding much of the low level detail.

¹ CANopen is a development of the Bosch CAN bus system, and is highly regarded among field-bus systems in terms of reliability and flexibility. It has excellent bus arbitration and error detection facilities, behaves well in sub-optimal electrical environments, as well as the ability to operate reliably at higher levels of bus utilisation.

3. SAFETY SYSTEM

In terms of “Safety” the enclosure has two, often conflicting, goals: protect the equipment inside from damage (by the elements; rain etc), and to prevent injury to any personnel inside. These enclosures are often deployed to remote and un-manned locations, so the need for some level of automation to protect the sensitive and expensive scientific equipment is obvious. This automation also means that the various parts of the enclosure (shutters and vent doors) are able to move without warning, which can be both surprising and extremely dangerous to any personnel inside (or working on the outside of) the enclosure.

The design of Ice Storm Series II safety system takes these factors into account.

The following feature help ensure that these goals are met:

- E-Stop buttons
- E-Close button
- Manual override controls for Shutters and Vent doors
- Several levels of lifeguard messages
- External inputs to indicate the need to close (UPS, rain sensors etc)
- “People Safe” switch and indicator
- Capability to install a siren for failure of PersonnelSafe conditions

3.1 Dome States

The Shutters, Vent doors and azimuth node all share a set of states known as the Dome States. These are as follows (in priority order):

- Fault – The application has determined that there is a fault. This will generally be something very serious, like a Drive Motor failure etc. The node will not accept any move commands from software.
- E-Stop – One of the E-Stop buttons or the software E-Stop has become active. The node will do all it can to stop as soon as is safely possible. In the case of the shutters this means immediately applying the brakes. The azimuth cannot stop this quickly due to the very large amount of mass that is being rotated – it will remove drive and coast to a halt. Once all the E-Stop inputs (button and software) become inactive a reset command needs to be sent to the Safety node to clear this state, until then the node will not accept any move commands from software.
- Manual Hardware – The manual override switch for this node is active. The node can only be controlled via the manual control box. In the case of the shutters the limit switches are ignored in this state. The node will not accept move commands from software.

- E-Close – The Safety node has put the bus into this mode when the E-Close button (or a software E-Close) has become active, the node will try to ‘close’ if possible. For the vent doors and shutter this means actually moving to the close/home position. Once the E-Close inputs (button and software) become inactive a reset command needs to be sent to the Safety node to clear this state, until then the node will not accept any move commands from software.
- Personnel Safe – The Safety node has put the bus into this state. This state also involves several components from the high level software, and these will provide IP address filtering to prevent commands from ‘remote’ agents. In this state the enclosure’s moving parts will not move unless commanded to do so by someone inside the enclosure.
- Manual Software – This state is entered into via a software command. The shutters in this state will allow step moves via software (without being homed), and will also ignore end of travel limits.
- E-Secure – This is entered into because some input to the Safety node has become active, and stayed active for its predefined time. For example the UPS has indicated that it is running on mains power for the last 5 minutes. Once entered into this state is the same as E-Close. The only difference is the priority – note that E-Secure has a lower priority than Personnel Safe, and E-Close a higher. The Safety node has put the bus into this state.
- Autonomous – The ‘default’ state when nothing else is active. In this state the enclosure is able to accept commands via software and is also able to autonomously move (close) or stop due to various other inputs or timeouts (watchdogs). It can also enter any of the other states.

Node Inputs							
Fault	E-Stop	Manual Drive Key	E-Close	Personnel Safe Key	Software Manual	E-Secure	State
Active	Don't Care	Don't Care	Don't Care	Don't Care	Don't Care	Don't Care	Fault
Inactive	Active	Don't Care	Don't Care	Don't Care	Don't Care	Don't Care	E-Stop
Inactive	Inactive	Active	Don't Care	Don't Care	Don't Care	Don't Care	Manual H/W
Inactive	Inactive	Inactive	Active	Don't Care	Don't Care	Don't Care	E-Close
Inactive	Inactive	Inactive	Inactive	Active	Don't Care	Don't Care	Personnel Safe
Inactive	Inactive	Inactive	Inactive	Inactive	Active	Don't Care	Manual S/W
Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Active	E-Secure
Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Inactive	Autonomous

Table 1 - Dome State priority table

Table 1 displays the priority of the states. As we can see, if the Fault input is active then the device will be in the Fault state regardless of the other inputs. Using this table we can determine which state a device will be in with a given set of inputs. For example we can see that if a device is the Manual Hardware state and an E-Stop button is pressed that the E-Stop has a higher priority and therefore the device will enter the E-Stop and stop moving. We can also see that if the device is in Manual Hardware and an E-Close occurs that the device will stay in the Manual Hardware state.

3.2 Basic States

All of the nodes on the electro-mechanical bus share the following simple states:

Node State	Dome Framework State	Description
INIT	INIT	Dome Nodes can initialise in this state
OPERATE	OPERATING MANUAL H/W	Dome Nodes are to operate only by manual H/W controls.
	OPERATING MANUAL S/W	Dome Nodes are to be controlled manually by S/W interface.
	OPERATING PERSONNEL SAFE	Dome Nodes must only be driven by commands in this state. Autonomous behaviour must not be carried out.
	OPERATING AUTONOMOUS	Dome Nodes can be driven by commands in this state, and autonomous behaviour is allowed.
EMERGENCY	CLOSED	Dome Nodes must close and must remain closed in this state.
	STOPPED	Dome Nodes must stop and must remain stopped in this state.
	SECURED	Dome Nodes must secure themselves and must remain safe in this state.
FAULT	IN_FAULT	Dome Nodes are in fault in this state.

Table 2 - basic state table

3.3 Lifeguard / Watchdog

There are three types of lifeguard messages implemented.

- NodeGaurd – between the CAN bus controller (CAN Server) and the node. This allows the node to know whether the CAN Server is running and also whether it is an active participant on the CAN bus.
- AppLifeLine – between the controlling application and the node. This allows the node to know whether it is being controlled high level software.
- SRDO – between the Safety node and the node. This allows the node to know whether it can ‘hear’ the Safety node. The Safety node is responsible for propagating the E-Close, E-Secure and Personnel Safe status over the CAN bus as well as controlling E-Stop input that is hardwired to each node.

Each of these lifeguard messages can be in one of the following states:

- Present – it is arriving on time and is correctly formatted
- Broken – was Present and has timed out
- Waiting – are expecting to be used, but have not received the first message yet.
- Disable – not being used.

The relationship between the Lifeguard states and the Dome states is defined in the following table:

Dome Node States									
Lifeline		States							
NODE	APP	E-STOP	FAULT	MANUAL H/W DRIVE	E-CLOSE	PERSONNEL SAFE	MANUAL S/W DRIVE	E-SECURE	AUTONOMOUS MODE
PRESENT	PRESENT	Stopped	In Fault	Operating H/W Manual	Closed	Operating Personnel Safe	Operating S/W Manual	Secured	Operating Autonomous
PRESENT	BROKEN	Stopped	In Fault	Operating H/W Manual	Closed	Stopped	Stopped	Secured	Closed
PRESENT	DISABLED	Stopped	In Fault	Operating H/W Manual	Closed	Operating Personnel Safe	Operating S/W Manual	Secured	Operating Autonomous
BROKEN	PRESENT	Stopped	In Fault	Operating H/W Manual	Closed	Stopped	Stopped	Secured	Closed
BROKEN	BROKEN	Stopped	In Fault	Operating H/W Manual	Closed	Stopped	Stopped	Secured	Closed
BROKEN	DISABLED	Stopped	In Fault	Operating H/W Manual	Closed	Stopped	Stopped	Secured	Closed
DISABLED	PRESENT	Stopped	In Fault	Operating H/W Manual	Closed	Operating Personnel Safe	Operating S/W Manual	Secured	Operating Autonomous
DISABLED	BROKEN	Stopped	In Fault	Operating H/W Manual	Closed	Stopped	Stopped	Secured	Closed
DISABLED	DISABLED	Stopped	In Fault	Operating H/W Manual	Closed	Operating Personnel Safe	Operating S/W Manual	Secured	Operating Autonomous

Table 3 - Lifeline state table

3.4 Personnel Safe State

The aim of Personnel Safe state is to ensure that the only movement that will occur is as a result of some action that a person inside the dome has instigated. In practice this means turning off the autonomous (equipment protecting) movement, and limiting the number of places software commands can be issued. This limiting is implemented by restricting the IP addresses that various servers will accept commands from, ideally to only machines physically located inside the enclosure. As the dome is co-rotating it is best if Telescope motion is also limited, thus IP address filtering occurs on several servers, including: DomeServer; TelescopeServer; and the low level CANServers(s).

In order to minimize the coupling between the software component the DomeServer is not responsible for the co-ordination of the PersonnelSafe state. There is a PeopleSafeServer that is responsible for monitoring the PersonnelSafe input switch and taking the appropriate action when it see a change.

Personnel Safe state is entered into when the PeopleSafeServer recognises that the Personnel Safe switch has been activated. The server will put the low-level devices on the CAN bus into Personnel Safe state by issuing a command to the Safety node, and it will inform the various servers to apply a set of IP addresses as a filter.

When the PeopleSafeServer recognizes that the key has become active it will start to flash the light inside the switch in order to indicate that it has seen the switch become active and is attempting to put the enclosure into Personnel Safe state. It will issue the appropriate commands, and wait for all the relevant devices (shutters, azimuth and vent doors) to enter people safe state. Only when everything is in the correct state will the server make the light stop flashing and stay on.

The PeopleSafeServer will continuously monitor the state of the dome and will detect if any of the devices are no longer in Personnel Safe state. At present the only action that will occur as a result of this is that the light will again flash. It is possible (and also safer) to make some sort of siren sound or light flash etc. to notify any personnel that the dome is no longer in Personnel Safe state.

When the PeopleSafeServer detects that the key has become inactive it commands the various servers to remove the IP address filtering and commands the Safety node to remove the PersonnelSafe input. The devices will the transition in to the next most appropriate state, see Table 1 Table 1 - Dome State priority table for details.

An issue with the current implementation is that modern networking bandwidth and tools like Remote Desktop and VNC allow a remote user to easily take control of machines that are inside the dome and issue commands as if they were standing in front of the local machine. There is currently nothing to prevent this. It is strongly recommended that the machines that will be used inside the dome are not remotely accessed for normal operations.

3.5 E-Secure Inputs

The Safety node provides multiple inputs that may be configured to be E-Secure inputs. Each of these may have a configurable internal timer attached to it so that if the input becomes active the E-Secure state is held off until the timer has counted down to zero (0). There are software commands to reset these timers.

As an example, most enclosures will come with a UPS used to ensure that the dome can close in the event of a mains power failure. The UPS will provide a digital output indicating whether it is running on mains or battery power. This will be connected into an input to the Safety node and configured to be active when the UPS is on battery power and the count-down timer for this input configured to be 60 seconds. If the mains power fails then without the intervention of higher level software the dome will close after one minute. If the higher level software needs more time (perhaps it knows that the mains will be back in two minutes and does not want to waste time closing and the opening the dome) it may send a command to reset the timer. By doing so the higher level software is taking responsibility for battery life and ensuring that the UPS has enough power remaining to close the dome.

There is no software access to these low-level count-down timers through the Dome Server, and details on how to modify these timers are outside the scope of this document. It is strongly recommended that these timers are not modified by anyone without first consulting EOS for further advice.

4. SCHEMA

This section describes the detailed syntax of the IceStorm Enclosure schema and refers to Release 0.10 of the schema.

The schema for the IceStorm Enclosure is shown below (for a detailed description of individual fields within the schema, see FIELD Definitions).

```

/*****
*****

/* This is so that the Server can validate the Schema Version. */
__Schema_Version__
{
    __Schema_Version_Major__0 INTEGER
    __Schema_Version_Minor__10 INTEGER
}

/* These version/info structs are published and can be requested */
ApplicationInfoStruct
{
    Version        STRING        /* Application version information */
    BuildInfo      STRING        /* Application build information */
}

SchemaVersionStruct
{
    SchemaVersionMajor INTEGER    /* Schema major version */
    SchemaVersionMinor INTEGER   /* Schema minor version */
}

VentDoor
{
    Commands STRUCTURE
    {
        /* Cause the door to start moving open.  If already open does nothing */
        Open STRUCTURE
        {
            Parameters STRUCTURE {}
            Returns STRUCTURE {}
        }
        /* Cause the door to start moving closed.  If already closed does nothing */
        Close STRUCTURE
        {
            Parameters STRUCTURE {}
            Returns STRUCTURE {}
        }
        /* If moving will cause the door to stop and the status to go to Stopped.
        Otherwise does nothing */
        Stop STRUCTURE
        {
            Parameters STRUCTURE {}
            Returns STRUCTURE {}
        }
    }

    Attributes STRUCTURE
    {
        /* Will be one of:
        VENT_STOPPED = 1,
        VENT_OPENED = 2,
        VENT_CLOSED = 3,
        VENT_MOVING_OPEN = 4,
        VENT_MOVING_CLOSE = 5,
        VENT_FAULT = 6
        */
        State INTEGER

        /* will be one of :
        VENT_STOPPED = 1,
        VENT_OPENED = 2,
        VENT_CLOSED = 3,
        VENT_FAULT = 6
        */
    }
}

```



```

Limits INTEGER

/* Dome state is one of:
   DOME_STATE_INIT           = 0
   DOME_STATE_OPMANHW       = 1
   DOME_STATE_OPMANSW       = 2
   DOME_STATE_PERSONSAFE    = 3
   DOME_STATE_OPAUTO        = 4
   DOME_STATE_CLOSED        = 5
   DOME_STATE_STOPPED       = 6
   DOME_STATE_SECURED       = 7
   DOME_STATE_INFAULT       = 8
*/
DomeState INTEGER

/* This is common for all Vents, Azimuth and Shutters */
NodeStatus STRUCTURE
{
    /* All NodeStatus.NodeState values are one of:
       APP_INIT               = 0
       APP_OPERATE            = 1
       APP_EMERGENCY          = 2
       APP_FAULT              = 3
    */

    NodeState INTEGER

    /* All lifeline values are one of:
       LIFELINE_PRESENT       = 0 // Enabled and available
       LIFELINE_BROKEN        = 1 // Enabled and broken
       LIFELINE_WAITING       = 2 // Enabled and waiting to start
       LIFELINE_DISABLED      = 3 // Disabled
    */

    NodeLifeLine INTEGER
    AppLifeLine INTEGER
}

Faults STRUCTURE
{
    NoCommunications BOOLEAN
    Timeout BOOLEAN /* A move did not reach the limit in time */
    LimitSwitches BOOLEAN /* Limit switches are inconsistent - both on */
    UnexpectedChange BOOLEAN /* Limit switch changed state unexpectedly */
}
} /* VentDoor STRUCTURE */

DigitalOutputs
{
    Commands STRUCTURE
    {
        /* Cause to go 'Active' */
        Enable STRUCTURE
        {
            Parameters STRUCTURE {}
            Returns STRUCTURE {}
        }
        /* Cause to go 'Inactive' */
        Disable STRUCTURE
        {
            Parameters STRUCTURE {}
            Returns STRUCTURE {}
        }
    }
}

```

```

Attributes STRUCTURE
{
    OutputEnabled BOOLEAN

    Faults STRUCTURE
    {
        NoCommunications BOOLEAN
    }
}

CoolingSystem
{
    Commands STRUCTURE
    {
        /* Turn the cooling system On. This will cause the unit to attempt
        to maintain the SetPoint temperature. Will be under LoopController
        control */
        Enable STRUCTURE
        {
            Parameters STRUCTURE {}
            Returns STRUCTURE {}
        }
        /* Turn the cooling system Off. This will stop all temperature control.
        Fans off, chillers off */
        Disable STRUCTURE
        {
            Parameters STRUCTURE {}
            Returns STRUCTURE {}
        }
        /* Set the desired temperature. This is the temperature the LoopController
        will attempt to maintain if the unit is enabled. */
        SetSetPoint STRUCTURE
        {
            Parameters STRUCTURE
            {
                Temperature FLOAT /* In Deg C */
            }
            Returns STRUCTURE {}
        }
    }
}

Attributes STRUCTURE
{
    Enabled BOOLEAN

    /* The temperature that the User has asked be maintained */
    /* UserSetPoint FLOAT */

    /* Value that the controller is trying to achieve. This may differ from the
    UserSetPoint due to condensation (or other) concerns */
    /* InternalTarget FLOAT */

    /* This is the temperature that the Loop Controller is reporting
    (and using as its input value). */
    CurrentTemperature FLOAT

    SetPointError FLOAT          /* UserSetPoint - CurrentTemperature */

    Output FLOAT

    /* InternalTargetError FLOAT */      /* InternalTarget - CurrentTemperature
*/

    Faults STRUCTURE
    {
        NoCommunications BOOLEAN
    }
}

```

```

    }
} /* CoolingSystem STRUCTURE */

Shutter
{
    Commands STRUCTURE
    {
        /* Perform a Homing sequence on the Shutter. Until the home position
        is established Move commands will be rejected.
        This command will NOT invalidate the current Homed status.
        On successful completion of the Homing sequence the Homed and HomeOK
        flags will be set to true. On failure the HomeFailed flag will be set
true.
        Starting a homing sequence will set the HomeOk and HomeFailed false. */
Home STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    /* Stop the shutter from moving. This is a controlled/profiled stop */
Stop STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    /* Enable the shutter drive and release the brake. */
Enable STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    /* Disable the shutter drive and apply the brake. */
Disable STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    /* Move the shutter to the commanded angle. All angles are in degrees
and are measured from the horizon. (ie horizon = 0; zenith = 90 .
This command is only valid if the Homed flag is true. It may be issued
while the Shutter is in On, Moving, Stopping, Homing and,
if the DisableTimeout is none 0, the Off states.

    */
Move STRUCTURE
    {
        Parameters STRUCTURE
        {
            Angle FLOAT
        }
        Returns STRUCTURE {}
    }

    /* The Shuttters will automatically enable, when given a move or home
command, then move, then once in the new position (or homed) and
this timer counts down to 0 the shutter will be disabled.
If this value is set to 0 then the automatic enable/disable of the
shutter will be disabled */
SetDisableTimeout STRUCTURE
    {
        Parameters STRUCTURE
        {
            seconds INTEGER
        }
        Returns STRUCTURE {}
    }
GetDisableTimer STRUCTURE

```

```

    {
        Parameters STRUCTURE {}
        Returns STRUCTURE
        {
            seconds INTEGER
        }
    }
}
Attributes STRUCTURE
{
    /* Will be one of :
    INVALID      -1
    OFF          0
    ON           1
    STOPPING    2
    HOMING       3
    MOVING       4
    STEPPING    5
    SPEEDING    6
    FAULT       7
    */
    State INTEGER
    /* String version of the above - used for debugging with the DeviceBrowser and
    may be removed. */
    StateString STRING

    Enabled BOOLEAN /* If true then the Shutter is enabled, brakes
released */
    InPosition BOOLEAN /* Shutter is in the position it has been commanded to
be and is not moving */
    Homed BOOLEAN /* Shutter knows where the Home position is */
    HomeOk BOOLEAN /* The last Home command was successful */
    HomeFailed BOOLEAN /* The last Home command failed */
    AtHome BOOLEAN /* Shutters home switch is active */
    FrontLimit BOOLEAN /* At the Front position limit - ie 'smallest' angle
the shutter can reach */
    BackLimit BOOLEAN /* At the Back position limitie - ie 'largest' angle
the shutter can reach */
    ShuttersTouching BOOLEAN /* The shutters are touching */
    FrontSector BOOLEAN /* The FRONT shutter is in the Front sector */

    CommandedPosition FLOAT /* Position the Shutter has been commanded to */
    CurrentPosition FLOAT /* Current Position of the Shutter */

    DomeState INTEGER
    NodeStatus STRUCTURE
    {
        NodeState INTEGER
        NodeLifeLine INTEGER
        AppLifeLine INTEGER
    }

    Faults STRUCTURE
    {
        NoCommunications BOOLEAN
        /* To be decided */
    }

    DriveCurrent FLOAT /* Drive Current (Amps) - will come from actual
Drive */
}
}
DeviceServer
{
    State STRUCTURE
    {
        DeviceServerState STRING
    }
}

```

```

/* Attributes that belong to the server itself
(rather than individual devices within the server) */
Attributes STRUCTURE
{
    ApplicationInfo STRUCTURE ApplicationInfoStruct
    SchemaVersion    STRUCTURE SchemaVersionStruct
}

/* All of the commands that apply to the server
itself (rather than individual devices) */
Commands STRUCTURE
{
    /* Reset the server, e.g. to recover from the Error state */
    /* NOTE: We are still deciding on exact behaviour for this command */
    Reset STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    /* This will send a "setResolveFaults" to all Vents, Az and Shutters */
    ResolveFaults STRUCTURE
    {
        Parameters STRUCTURE
        {
            Device STRING
        }
        Returns STRUCTURE {}
    }

    /* Return version information applicable to the Server */
    GetVersion STRUCTURE
    {
        Parameters STRUCTURE
        {
        }
        Returns STRUCTURE
        {
            ApplicationInfo STRUCTURE ApplicationInfoStruct
            SchemaVersion    STRUCTURE SchemaVersionStruct
        }
    }

    /* This command gets passed through to the various dome devices.
the devices have thier own behaviour for when this LifeLine times out */
    RestartLifeLineTimer STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
}

/* The schema definitions for all of the devices
owned by the server */
Devices STRUCTURE
{
    Shutters STRUCTURE
    {
        /* This is purely a 'convenience' device; it is an amalamation of
both front and rear shutter */
        Commands STRUCTURE
        {
            Home STRUCTURE
            {
                Parameters STRUCTURE {}
                Returns STRUCTURE {}
            }
            Stop STRUCTURE
            {

```

```

        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    Enable STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    Disable STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    Move STRUCTURE
    {
        Parameters STRUCTURE
        {
            FrontAngle FLOAT
            RearAngle FLOAT
        }
        Returns STRUCTURE {}
    }
}
Attributes STRUCTURE
{
    Enabled BOOLEAN           /* Logical AND of both shutters */
    InPosition BOOLEAN       /* Logical AND of both shutters */
    Homed BOOLEAN            /* Logical AND of both shutters */
    Closed BOOLEAN           /* Logical AND of both shutters */
}
} /* Shutters STRUCTURE */

FrontShutter STRUCTURE Shutter
RearShutter STRUCTURE Shutter

Azimuth STRUCTURE
{
    Commands STRUCTURE
    {
        /* Make the Dome follow the Telescope */
        Enable STRUCTURE
        {
            Parameters STRUCTURE {}
            Returns STRUCTURE {}
        }
        /* Stop the Dome from moving - will not follow the Telescope */
        Disable STRUCTURE
        {
            Parameters STRUCTURE {}
            Returns STRUCTURE {}
        }
    }
}

Attributes STRUCTURE
{
    /* No 'real' units. Will be an analog value converted to digital
    and then have a scale and offset applied. The 'units' will
    depend on the scale and offset */
    OffsetAngle FLOAT

    /* Will be one of:
    AZIMUTH_DISABLED = 1,
    AZIMUTH_ESTOP = 2,
    AZIMUTH_READY_OP = 3,
    AZIMUTH_ALIGN_TSC = 4,
    AZIMUTH_RUN = 5,
    AZIMUTH_EOT_RECOVERY = 6,

```

```

        AZIMUTH_FAULT = 7
        AZIMUTH_RESET_DRV_FAULT = 8
    */

    Status INTEGER

    Inputs STRUCTURE
    {
        CCWTravelLimit BOOLEAN
        CWTravelLimit BOOLEAN
        CCWSofLimit BOOLEAN
        CWSofLimit BOOLEAN
        CCWHardLimit BOOLEAN
        CWHardLimit BOOLEAN
        /* The End Of Travel Override key - true = active */
        EOTOverRideKey BOOLEAN
    }
/*
    Outputs STRUCTURE
    {
        TelescopeEnabled BOOLEAN
        DriveEnabled BOOLEAN
        BrakeHoldOff BOOLEAN
        HardLimitOverride BOOLEAN
    }
*/

    DomeState INTEGER

    NodeStatus STRUCTURE
    {
        NodeState INTEGER
        NodeLifeLine INTEGER
        AppLifeLine INTEGER
    }

    Faults STRUCTURE
    {
        NoCommunications BOOLEAN /* This Server cannot
communicate with the Azimuth device */
        InconsistentLimitSwitches BOOLEAN
        MotorDriveTemperatureMaster BOOLEAN /* If true then this drive
is in an over */
        MotorDriveTemperatureSlave BOOLEAN /* temperature state
*/
        MotorDriveNotReadyMaster BOOLEAN /* The Drive is reporting
*/
        MotorDriveNotReadySlave BOOLEAN /* that it is not ready
*/
    }
    }

} /* Azimuth STURCTURE */

/* VentDoors */
VentDoor1 STRUCTURE VentDoor
VentDoor2 STRUCTURE VentDoor
VentDoor3 STRUCTURE VentDoor
VentDoor4 STRUCTURE VentDoor

/* CoolingSystems */
FanCoolingUnit1 STRUCTURE CoolingSystem
FanCoolingUnit2 STRUCTURE CoolingSystem
FanCoolingUnit3 STRUCTURE CoolingSystem

HouseLights STRUCTURE
{

```

```

/* All that can be done is to issue a command to turn off
the lights. There is no feedback to indicate current state */
Commands STRUCTURE
{
    /* Will issue a pulse to the lighting controller
to turn the lights off. Does not keep them off */
TurnOff STRUCTURE
{
    Parameters STRUCTURE {}
    Returns STRUCTURE {}
}
TurnOn STRUCTURE
{
    Parameters STRUCTURE {}
    Returns STRUCTURE {}
}
}
Attributes STRUCTURE
{
    Alloff BOOLEAN

    Faults STRUCTURE
    {
        NoCommunications BOOLEAN
    }
}
}

/* General Dgital Outputs */
/* HouseLights STRUCTURE DigitalOutputs */

SnowMelt STRUCTURE DigitalOutputs
RecirculationFan1 STRUCTURE DigitalOutputs
RecirculationFan2 STRUCTURE DigitalOutputs

Safety STRUCTURE
{
    Commands STRUCTURE
    {
        /* Software command to enter into E-Stop state */
SetSWESTop STRUCTURE
{
    Parameters STRUCTURE {}
    Returns STRUCTURE {}
}
SetSWEClose STRUCTURE
{
    Parameters STRUCTURE {}
    Returns STRUCTURE {}
}
SetSWESecure STRUCTURE
{
    Parameters STRUCTURE {}
    Returns STRUCTURE {}
}
ClearSWESTop STRUCTURE
{
    Parameters STRUCTURE {}
    Returns STRUCTURE {}
}
ClearSWEClose STRUCTURE
{
    Parameters STRUCTURE {}
    Returns STRUCTURE {}
}
ClearSWESecure STRUCTURE
{

```



```

        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    /* The E * conditions are all latched. Thus the reset is required
    to recover from these even if the Button is no longer active */
    ResetEStop STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    ResetEClose STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    ResetESecure STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }
    /* The Safety Node can issue a Secure command to all nodes on the
    bus. This will cause all nodes to go into the Secure state.
    Various inputs (like the UPS) will want a delay between when
    they go active till when this command is issued. Once the
    input becomes active these timers will start counting down.
    Once any of them reach Zero then the Secure command will
    be issued to the bus. This command will restart the all of
    these count down timers. */
    /* Note that in the case of PS1 there is only the UPS that triggers
    a delayed ESecure
    In other cases there may/will be others - rain/dust/cloud/
    smoke sensors etc */
    ESecureHoldOff STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE {}
    }

    SetUPSHoldOffTime STRUCTURE
    {
        Parameters STRUCTURE
        {
            Seconds FLOAT
        }
        Returns STRUCTURE {}
    }
    GetUPSHoldOffTime STRUCTURE
    {
        Parameters STRUCTURE {}
        Returns STRUCTURE
        {
            Seconds FLOAT
        }
    }
}

Attributes STRUCTURE
{
    /* Inputs that will cause the Dome to E-Stop */
    StopInputs STRUCTURE
    {
        SoftwareEStop BOOLEAN
        EStop1Button BOOLEAN
        EStop2Button BOOLEAN
        EStop3Button BOOLEAN
        EStop4Button BOOLEAN
        EStop5Button BOOLEAN
        EStop6Button BOOLEAN
    }
}

```

```
    }

    /* Inputs that will cause the Dome to E-Close */
    CloseInputs STRUCTURE
    {
        SoftwareEClose BOOLEAN
        ECloseButton BOOLEAN
    }

    /* Inputs that will cause the Dome to E-Secure */
    SecureInputs STRUCTURE
    {
        SoftwareESecure BOOLEAN

        /* Delayed Inputs for E-Secure */
        UPSInput BOOLEAN
    }

    /* Current OUTPUT values */
    EStopState BOOLEAN
    ECloseState BOOLEAN
    ESecureState BOOLEAN

    /* This is the number of seconds until the E-Secure output
       will be set. This is the lowest of any/all the count down
       timers described above */
    /* If none of the E-Secure input lines are active this will be NULL */
    ESecureHoldOffTime INTEGER

    Faults STRUCTURE
    {
        NoCommunications BOOLEAN
    }
} /* Safety sub-system */
}
```

5. FIELD DEFINITIONS

This section describes the individual fields within the schema. The field descriptions are divided into separate sections for each device within the server.

The following sections will use a standard layout to describe the various DML options. A Command will take the basic form of:

CommandName

Description:

Describes the command's function and requirements.

Parameters:

none or a description of the parameters

Returns:

This will describe the DML that a command may return. If the command does not return any DML then "none" will be used. This does not mean that the command does not return anything. ALL commands will return the normal DMLResult (see the API documentation) being one of: OK, Rejected, or Failed. If the result is not OK then a textual description will be returned.

Example:

This will include example usage, including what may be returned.

An attribute will have the basic form of:

AttributeName

Description:

Describes the attribute's meaning and expected values.

DML Type:

One of BOOLEAN, INTEGER, STRING, FLOAT, BINARY or STRUCTURE. If STRUCTURE then a description of each field will be included.

Example:



5.1 Server

This section describes the schema fields for the server itself (as distinct from the individual devices owned by the server).

5.1.1 Commands

This section describes the commands at the DeviceServer level, i.e. DeviceServer.Commands.*

Reset

Description:

The server will send this command to all devices on the bus. It is used to recover a device from a "Fault" state.

Parameters: none

Returns: none

Example: DeviceServer.Commands.Reset {}

ResolveFaults

Description:

This command is also used to recover a device from an error/fault condition, however is slightly more lightweight, and can be targeted to a specific device.

Parameters:

Parameters.Device as a STRING

This (optional) parameter specifies which device (FrontShutter; RearShutter; Azimuth; VentDoor1-4 etc) to send a ResolveFaults command to. If not set then the command will be sent to all devices.

Returns: none

Example:

DeviceServer.Commands.ResolveFaults {} OR

DeviceServer.Commands.ResolveFaults.Parameters {Device "FrontShutter" }

GetVersion

Description:

Returns version information for the server.

Parameters: none

Returns:

ApplicationInfo

DML Type: STRUCTURE

Version

The version number of the application. This will typically be a three digit number representing the major, minor and release numbers.

DML Type: STRING

BuildInfo

Not yet implemented.

Description: STRING

SchemaVersion

DML Type: STRUCTURE

SchemaVersionMajor

The major version number of the schema. The number will change between releases if there have been significant changes.

DML Type: INTEGER

SchemaVersionMinor

The major version number of the schema. The number will change between releases if there have been in-significant changes.

DML Type: INTEGER

Example:

Send DeviceServer.Commands.GetVersion {}

With a return of

DeviceServer.Commands.GetVersion { Returns {

 ApplicationInfo { Version "2.1.0" }

 SchemaVersion { SchemaVersionMajor 0 SchemaVersionMinor 10 }

} }

RestartLifeLineTimer

Description:

Resets the application lifeline timer of all the devices on the CAN bus. If the lifeline timers are not reset within the set timeout period, the devices will generally either stop or close depending on their current state (see Safety System above).

The value of these LifeLineTimers is set at a low level and it outside the scope of this document.

Parameters: none**Returns:** none**Example:** `DeviceServer.Commands.RestartLifeLineTimer {}`

5.1.2 Attributes

ApplicationInfo

Description:

See the return of the GetVersion command.

DML Type: STRUCTURE

Example: DeviceServer.Attributes.ApplicationInfo { Version "2.1.0" }

SchemaVersion

Description:

See the return of the GetVersion command.

DML Type: STRUCTURE

Example: DeviceServer.Attributes.SchemaVersion {

 SchemaVersionMajor 0

 SchemaVersionMinor 10

}

5.2 Common/Shared among Multiple Devices

5.2.1 Attributes

The Azimuth, both Shutters, and all the Vent doors publish the following attributes:

DomeState

Description:

Current State of the device. Will be one of:

Name	Value
Init	0
Manual Hardware	1
Manual Software	2
Personnel Safe	3
Autonomous	4
E-Close	5
E-Stop	6
E-Secure	7
Fault	8

DML Type: INTEGER

Example: DeviceServer.Devices.FrontShutter.Attributes.DomeState 4

NodeState

Description:

Current State of the node. Will be one of:

Name	Value
Init	0
Operate	1
Emergency	2
Fault	3

DML Type: INTEGER

Example: DeviceServer.Devices.Azimuth.Attributes.NodeState 1

NodeLifeLine

Description:

Describes the current state of the CAN bus level and Application level lifelines. Will be one of:

Name	Value
Present	0
Broken	1
Waiting	2
Disabled	3

See the Safety section above for a more detailed description.

DML Type: INTEGER

Example: DeviceServer.Devices.VentDoor3.Attribibutes.NodeLifeLine 0

ALL of the devices publish a Faults STRUCTURE with at least the following:

Faults

Description:

The STRUCTURE to contain the various faults that may occur.

DML Type: STRUCTURE

NoCommunications

Description:

The server cannot communicate with the node/device. All devices will publish at least this filed in the Faults STRUCTURE, some will publish others.

DML Type: BOOLEAN

Example: DeviceServer.Devices.FrontShutter.Attributes.Faults.NoCommunications FALSE

5.3 Azimuth

This section describes the schema fields for the Azimuth sub-system. The Azimuth sub-system has the responsibility of aligning the azimuth of the dome, with the azimuth of the telescope.

5.3.1 Commands

Enable

Description:

Enables the azimuth sub-system to track the azimuth of the telescope, aligning the azimuth of the dome and telescope.

Once in the Run state, if the azimuth is unable to follow the telescope, initially a soft limit will be reached and the azimuth will remain operational until the hard limit is reached, causing the azimuth to enter Disabled state.

Parameters: none

Returns: none

Example: DeviceServer.Devices.Azimuth.Commands.Enable {}

Disable

Description:

Disables the azimuth sub-system; alignment between the telescope azimuth and the dome azimuth will no longer be maintained.

Parameters: none

Returns: none

Example: DeviceServer.Devices.Azimuth.Commands.Disable {}

5.3.2 Attributes

This device does have **DomeState** and **NodeStatus** attributes as described in 5.2 above.

OffsetAngle

Description:

Not implemented yet.

DML Type: FLOAT

Example: N/A

Status

Description:

Current status of the azimuth. Valid states of the azimuth are:

Value	Status	Description
1	Disabled	Startup state. The azimuth drives are off, some the dome will not follow the telescope. Will transition to ReadyOp when the Drive Motors indicate that they are ready.
2	EStop	An E-Stop if active. The azimuth will stop as quickly as it can and will stay stopped.
3	ReadyOp	Ready to begin operating – the Drive Motors are ready (but not actually enabled/running). If the End Of Travel limits are not active and the EOT Key switch override is not active then an enable command will transition to Align. If an EOT limit is active and the EOT Key switch is active then an enable command will transition to EOT Recovery.
4	Align	Enables the drives and will align the Dome with Telescope. Once/If there are no relative limits (soft or hard) then will transition to Run.
5	Run	Azimuth Drives are enabled. In this state the Dome allows the Telescope to move (in Azimuth).
6	End-Of-Travel recovery	Allows recovery from the Dome's absolute End-Of-Travel limits. Will stay in this state until the EOT Key is switched turned off.
7	Fault	This state is entered into whenever a 'Fault' is detected in any other state. These faults can be either an inconsistent set of switch inputs (e.g. both CW and CCW switches active), or a Drive fault (not ready or over temperature).
8	Drive Fault Recovery	Attempt to recover from a Drive fault. If successful transitions to Disabled, otherwise back to Fault.

DML Type: INTEGER

Example: DeviceServer.Devices.Azimuth.Attributes.Status 3

Inputs

Description:

The status of the inputs to the Azimuth system.

DML Type: STRUCTURE

This STRUCTURE contains the following fields:

CWTravelLimit and CCWTravelLimit

Description:

Status of the Clock-Wise and Counter-Clock-Wise absolute travel limits for the dome. This are considered **very** serious limits, and if activated the dome will perform an emergency stop.

DML Type: BOOLEAN

Example: DeviceServer.Devices.Azimuth.Attributes.Inputs.CWTravelLimit FALSE

CWSoftLimit and CCWSoftLimit

Description:

Status of the Clock-Wise and Counter-Clock-Wise relative soft limit between the telescope and the dome. If active then the dome will attempt to prevent further relative motion in this direction.

DML Type: BOOLEAN

Example: DeviceServer.Devices.Azimuth.Attributes.Inputs.CCWSoftLimit FALSE

CWHardLimit and CCWHardLimit

Description:

Status of the Clock-Wise and Counter-Clock-Wise relative hard limit between the telescope and the dome. The relative error between the telescope and dome has progressed through the soft limit into the hard limit. When this limit becomes active the system will transition into the Disabled state, and motion of both telescope and dome azimuth will cease.

DML Type: BOOLEAN

Example: DeviceServer.Devices.Azimuth.Attributes.Inputs.CCWHardLimit TRUE

EOTOverrideKey

Description:

The End-Of-Travel override key status. The key is used to recover from the dome's absolute EOT limits.

DML Type: BOOLEAN

Example: DeviceServer.Devices.Azimuth.Attributes.Inputs.EOTOverrideKey TRUE

Example:

```
DeviceServer.Devices.Azimuth.Attributes.Inputs {  
    CCWTravelLimit FALSE  
    CWTravelLimit FALSE  
    CCWSoftLimit FALSE  
    CWSoftLimit FALSE  
    CCWHardLimit FALSE  
    CWHardLimit FALSE  
    EOTOverrideKey FALSE  
}
```

Faults

Description:

Fault status of the Azimuth node.

DML Type: STRUCTURE

InconsistentLimitSwitches

Description:

True if any of the limit switches are in a state that is not physically possible, meaning that there is an error with a switch value. An example is if both the CW and CCW EOT are active.

DML Type: BOOLEAN**Example:** DeviceServer.Devices.Azimuth.Attributes.Faults.InconsistentLimitSwitches

FALSE

MotorDriveTemperatureMaster and MotorDriveTemperatureSlave

Description:

True if the Motor Drive (Master or Slave) is in an over temperature condition.

DML Type: BOOLEAN**Example:**

DeviceServer.Devices.Azimuth.Attributes.Faults.MotorDriveTemperatureMaster

FALSE

MotorDriveNotReadyMaster and MotorDriveNotReadySlave

Description:

True if the Motor Drive (Master or Slave) is in a Not Ready condition, meaning that it will not be able to be enabled or moved.

DML Type: BOOLEAN**Example:** DeviceServer.Devices.Azimuth.Attributes.Faults.MotorDriveNotReadyMaster

FALSE

Example:

```
DeviceServer.Devices.Azimuth.Attributes.Faults {  
    NoCommunications FALSE  
    MotorDriveTemperatureMaster FALSE  
    MotorDriveTemperatureSlave FALSE  
    MotorDriveNotReadyMaster FALSE  
    MotorDriveNotReadySlave FALSE  
}
```

5.4 Shutter

This section describes the schema fields for the Shutter sub-system. The Shutter sub-system can be treated as a combined system where certain commands are applied to both shutters or commands can be issued to a specific shutter. The two individual shutters are known as 'FrontShutter' and 'RearShutter'.

5.4.1 Individual Shutter

This section describes the commands and attributes used for an individual shutter.

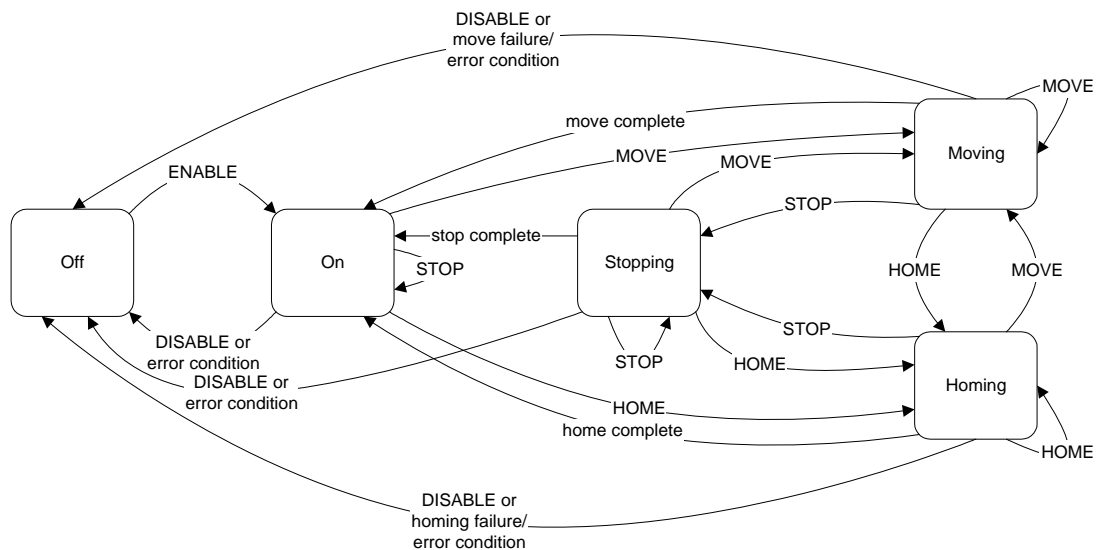


Figure 1 - Shutter state machine

5.4.1.1 Commands

Home

Description:

This command is used to home an individual shutter.

Homing of a shutter will enable it to find its absolute reference point. An absolute reference point must be established before a shutter can be commanded to a given position.

If the shutter has previously been homed this command will not force it to forget the previous home position until a new position has been found.

If **both** shutters are sealed (i.e. both are at home, and the shutters touching status is true²) when this command is received the shutter will use the current position as home without actually moving to find a new home position. This is so that if the shutters power up they can establish home (assuming they are home) without having to move and become un-sealed.

The State diagram for the shutters above (as distinct from the general Dome level states; E-Stop, Autonomous etc.) shows which States this command may be issued from. If the shutter is in an invalid state (e.g. Off) this command will be rejected.

This command is only valid from the Dome Autonomous and PersonnelSafe states and will be rejected for all others.

The DomeServer will also auto-enable the shutter if it is configured to do so (see SetDisableTimeout command below).

Parameters: none

Returns: none

Example: DeviceServer.Devices.FrontShutter.Commands.Home {}

Stop

Description:

This command is used to stop the motion of a shutter. The shutter will come to a controlled (profiled) stop. It will continue to servo at the stopped position, i.e. the motor will still be enabled and the brakes released.

Parameters: none

Returns: none

² Due to the mechanical design of the shutters, if they are both at their home switch **and** they are touching, then this provides a precise enough absolute position indication to be used for further relative moves. A 'normal' Home sequence where the shutter moves and finds the edge of the home switch is still considered slightly more accurate.

Example: DeviceServer.Devices.RearShutter.Commands.Stop {}

Move

Description:

This command is used to move a shutter to a desired angle. It will be rejected if

- the Angle parameter is not supplied
- the shutter is not in an appropriate State
- the shutter has not been homed
- the commanded position is outside of the range of the shutters
- the move may cause the shutters to collide the command.

The State diagram for the shutters above (as distinct from the general Dome level states; E-Stop, Autonomous etc.) shows which States this command may be issued from. If the shutter is in an invalid state (e.g. Off) this command will be rejected.

This command is only valid from the Dome Autonomous and PersonnelSafe states and will be rejected for all others.

The DomeServer will also auto-enable the shutter if it is configured to do so (see SetDisableTimeout command below).

Parameters:**Parameters.Angle**

The requested angle to move to. The angle is measured from the 'front' horizon.

DML Type: FLOAT

Units: Degrees

Range: Front ~ 25° – 155 °

Rear ~ 58 ° – 155 °

Note that these figures are indicative only and vary slightly on a per site basis, depending on factors such as the exact switch and dog positions etc.

Returns: none

Example:

```
DeviceServer.Devices.RearShutter.Commands.Move { Parameters { Angle 100.5 } }
```

Enable

Description:

This command is used to enable the shutter drive. This will cause the shutter to enable power to the drive, release the brake and actively servo to maintain the current position.

Parameters: none

Returns: none

Example: DeviceServer.Devices.RearShutter.Commands.Enable {}

Disable

Description:

This command is used to disable the shutter drive. This will cause the shutter to apply the brake and (a short time later³) to disable the drive motor.

If the shutter is moving (e.g. from a Move or Home command) when this command is issued it will still immediately apply the brakes. This will result in an abrupt stop rather than the normal smooth deceleration; however the shutter is designed to withstand this.

Parameters: none

Returns: none

Example: DeviceServer.Devices.RearShutter.Commands.Disable {}

SetDisableTimeout

Description:

The low-level shutter node will not accept a move command if the shutter is in the Off state. The DisableTimeout allows the server to automatically enable the shutter before issuing a Move (or Home) command, and it will disable the shutter some seconds after the move has completed (it will actually disable it once the shutter has been in the On state for some seconds). A value of zero (0) will disable this feature.

Parameters:

Parameters.seconds

The number of seconds the shutter can be in the On state until the server will disable it.

DML Type: INTEGER

Units: seconds

Range: This is a 32bit signed integer (0 – 2147483647)

Returns: none

³ This short delay is to enable the mechanical braking mechanism to take hold before disabling the drive. If this is not done then the Shutter may experience a small 'drop' when disabling.

Example:

```
DeviceServer.Devices.RearShutter.Commands.SetDisableTimeout {  
    Parameters {  
        seconds 10  
    }  
}
```

GetDisableTimer**Description:**

Returns the current value being used for the DisableTimer

Parameters: none

Returns:**Returns.seconds**

The current value of this setting.

DML Type: INTEGER

Units: seconds

Example:

```
Send DeviceServer.Devices.RearShutter.Commands.GetDisableTimeout { }  
Receive  
DeviceServer.Devices.RearShutter.Commands.GetDisableTimeout {  
    Returns {  
        seconds 10  
    }  
}
```

5.4.1.2 Attributes

Each of the FrontShutter and RearShutter has **DomeState** and **NodeStatus** attributes as described in 5.2 above.

State

Description:

The current state of the shutter's internal state machine.

Value	State	Description
-1	Invalid	The server can not obtain the state from the node, or the state the node is returning is not a valid state, e.g. when the node is powered off.
0	Off	The node is off. The drive is disabled and the brake is applied.
1	On	The drive is enabled, and the brake is released. The drive is currently servo'ing, and maintaining the current position
2	Stopping	The node is currently performing a profiled stop. This is usually as a result of a stop command. This stop will stop as soon as the profile allows, rather than stopping at a pre-determined position.
3	Homing	The node is performing a homing sequence. Because the encoder mechanism is a relative system, the node needs to perform a homing sequence to determine an absolute reference point.
4	Moving	The node is performing a move.
5	Stepping	Not Implemented
6	Speeding	Not Implemented
7	Fault	Not Implemented

DML Type: INTEGER

Example: DeviceServer.Devices.RearShutter.Attributes.State 1

StateString

Description:

This is a string representation of the above attribute. Its main use is for debugging and it may be deprecated in a future version. The string value is the same as that in the State column in the table above.

DML Type: STRING

Example: DeviceServer.Devices.RearShutter.Attributes.StateString "On"

Enabled

Description:

The current enabled status on the shutter. True indicates that the drive is enabled and the brakes released.

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.Enabled FALSE

InPosition

Description:

Indicates whether the shutter is in the position it is supposed to be. This flag is especially useful when waiting for a move to complete.

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.InPosition FALSE

Homed

Description:

Indicates whether the shutter currently knows where its home position is. This **must** be true before a Move command will be accepted.

If this value is false then the CurrentPosition attribute is invalid.

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.Homed TRUE

HomeOk

Description:

True if the last Home command was successful. Note that even if the last Home command was not successful, it may still be possible that the Homed attribute (above) is true, and therefore subsequent Move commands will be accepted. However, in reality if a Home command fails there is a problem that needs to be rectified.

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.HomeOk TRUE

HomeFailed

Description:

Indicates if the last home command failed.

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.HomeFailed TRUE

AtHome

Description:

Indicates that the shutter has its Home switch active. It must be noted that this switch is active over a relatively large arc (several centimeters), so the actual status should not be used for any precise position indication.

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.AtHome TRUE

FrontLimit

Description:

The shutter has reached its front travel limit. If this occurs while the shutter is moving (Move or Home), then the shutter will immediately disable (go to Off state). If the shutter has a Home position (Homed is true) then it will be possible to enable the shutter again and command it to move out of the limit, i.e. to provide a larger angle than its current position. A command to drive it further into the limit will be rejected.

Under normal operations it should not be possible to command the shutter into the limit switches using software as there are software limits configured to be inside the range of hardware limit switches.

If this does occur please contact EOS for further advice.

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.FrontLimit TRUE

BackLimit

Description:

The shutter has reached its back travel limit. If this occurs while the shutter is moving (Move or Home), then the shutter will immediately disable (go to Off state). If the shutter has a Home position (Homed is true) then it will be possible to enable the shutter again and command it to move out of the limit, i.e. to provide a smaller angle than its current position. A command to drive it further into the limit will be rejected.

Under normal operations it should not be possible to command the shutter into the limit switches using software as there are software limits configured to be inside the range of hardware limit switches.

If this does occur please contact EOS for further advice.

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.BackLimit TRUE

ShuttersTouching

Description:

Indicates the status of the proximity switch between the shutters. This is especially useful during a homing sequence.

This attribute should have the same value for the front and rear shutter.⁴

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.ShutterTouching TRUE

⁴ Of course there may be a slight delay in the various software layers and one shutter will publish a new value before the other.

FrontSector

Description:

For both front and rear shutters this attribute indicates whether the front shutter is in the front section. As the rear shutter can not mechanically move to the front section this can be of some use in basic collision avoidance. The shutters use this information as part of the homing sequence.

As per the ShuttersTouching attribute this should be the same for both shutters.

DML Type: BOOLEAN

Example: DeviceServer.Devices.RearShutter.Attributes.FrontSection TRUE

CommandedPosition

Description:

The position that the shutter has been commanded to. This will not be valid on startup.

If a Stop command is issued this attribute will update during the profiled stop until the shutter stops. If a Disable command is issued (either directly or internally as a result of say and E-Stop) then this attribute will be updated with the current position of the shutter.

DML Type: FLOAT

Units: Degrees

Example: DeviceServer.Devices.RearShutter.Attributes.CommandedPosition 58.0

CurrentPosition

Description:

The current position of the shutter. This will not be valid until a Home has been successfully completed.

DML Type: FLOAT**Units:** Degrees (from the 'front' horizon)**Example:** DeviceServer.Devices.RearShutter.Attributes.CurrentPosition 58.0

DriveCurrent

Description:

Not yet implemented.

DML Type: FLOAT**Units:** Amps**Example:** N/A

5.4.2 Shutters

This section describes the commands applicable to both shutters together. This “virtual” device is for convenience when performing actions on both shutters together.

5.4.2.1 *Commands*

Home

Description:

Performs a Home on both shutters.

Parameters: none

Returns: none

Example: DeviceServer.Devices.Shutters.Commands.Home {}

Stop

Description:

Stops both shutters.

Parameters: none

Returns: none

Example: DeviceServer.Devices.Shutters.Commands.Stop {}

Enable

Description:

Enable both shutters.

Parameters: none**Returns:** none**Example:** DeviceServer.Devices.Shutters.Commands.Enable {}

Disable

Description:

Disable both shutters.

Parameters: none**Returns:** none**Example:** DeviceServer.Devices.Shutters.Commands.Disable {}

Move

Description:

Move both shutters. The server will determine which shutter to move first depending on the directions.

Parameters:

Parameters.FrontAngle and Parameters.RearAngle

DML Type: FLOAT (both)

Units: Degrees (from 'front' horizon)

Range: see the Move command for the single shutter above.

Returns: none

Example:

```
DeviceServer.Devices.Shutters.Commands.Move {  
    Parameters {  
        FrontAngle 25.0  
        RearAngle 155.0  
    }  
}
```

5.4.2.2 Attributes

Enabled

Description:

A logical AND of both shutters' enabled attributes.

DML Type: BOOLEAN

Example: DeviceServer.Devices.Shutters.Attributes.Enabled TRUE

InPosition

Description:

A logical AND of both shutters' InPosition attributes.

DML Type: BOOLEAN

Example: DeviceServer.Devices.Shutters.Attributes.InPosition FALSE

Homed

Description:

A logical AND of both shutters' Homed attributes.

DML Type: BOOLEAN

Example: DeviceServer.Devices.Shutters.Attributes.Homed TRUE

Closed

Description:

Indicates that both shutters are AtHome and that they are touching.

In this state the Shutters are considered closed and sealed.

DML Type: BOOLEAN

Example: DeviceServer.Devices.Shutters.Attributes.Closed TRUE

5.5 Vent Doors

This section describes the schema fields for the vent door sub-system. The vent door sub-system consists of four identical vent doors described by the *VentDoor* DML structure. The following commands and attributes apply equally to VentDoor1 through VentDoor4.

5.5.1 Commands

Open

Description:

Causes the vent door to start moving open. If it is already open it is ignored.

This is only valid in the Dome Autonomous and PersonnelSafe states.

Parameters: none

Returns: none

Example: DeviceServer.Devices.VentDoor1.Commands.Open {}

Close

Description:

Causes the vent door to start moving closed. If it is already closed this command is ignored.

This is only valid in the Dome Autonomous and PersonnelSafe states.

Parameters: none

Returns: none

Example: DeviceServer.Devices.VentDoor2.Commands.Close {}

Stop

Description:

Causes the vent door to stop moving.

This is only valid in the Dome Autonomous and PersonnelSafe states.

Parameters: none

Returns: none

Example: DeviceServer.Devices.VentDoor3.Commands.Stop {}

5.5.2 Attributes

Each of the Vent Doors has **DomeState** and **NodeStatus** attributes as described in 5.2 above.

State

Description:

Describes the internal state of the vent door. It may be one of:

Value	State	Description
1	Stopped	The vent door is not moving, and it is not at either the Opened or Closed limit switches, i.e. it is somewhere in between.
2	Opened	The door is stopped and at the Opened position.
3	Closed	The door is stopped and at the Closed position.
4	Moving Open	The door is currently moving towards the Opened position.
5	Moving Close	The door is currently moving towards the Closed position.
6	Fault	The door is in a Fault state. See the Faults attributes for further details.

DML Type: INTEGER

Example: DeviceServer.Devices.VentDoor3.Attributes.State 3

Limits

Description:

Describes the current status of the vent door's limit switches. This will be one of:

Value	State	Description
1	Stopped	Neither the Opened or Closed limit is active. It is assumed that the door is somewhere between these two.
2	Opened	The door is at the Opened limit switch.
3	Closed	The door is at the Closed limit switch..
6	Fault	The switches are in a state that is not possible, e.g. both Opened and Closed simultaneously.

DML Type: INTEGER

Example: DeviceServer.Devices.VentDoor3.Attributes.Limits 3

Faults

Description:

Describes the various faults.

DML Type: STRUCTURE

Timeout

Description:

The last Move command failed because the move did not finish in the required time limit. This might happen, for example, after an Open command if the door is currently Closed and the Opened switch is broken.

A server-level ResolveFault or Reset command will reset this node.

DML Type: BOOLEAN

Example: DeviceServer.Devices.VentDoor1.Attributes.Faults.Timeout TRUE

LimitSwitches

Description:

The limit switches are in a non-sensical state, e.g. both are active.

DML Type: BOOLEAN

Example: DeviceServer.Devices.VentDoor3.Attributes.Faults.LimitSwitches FALSE

UnexpectedChange

Description:

The limit switches changed in a way that was not expected. For example if an Open command caused the Closed switch to become active, or if the switches change state when the door is not supposed to be moving.

A server level ResolveFault or Reset command will reset this node.

DML Type: BOOLEAN

Example: DeviceServer.Devices.VentDoor2.Attributes.Faults.UnexpectedChange FALSE

Example:

```
DeviceServer.Devices.VentDoor2.Attributes. Faults. {  
    NoCommunications FALSE  
    Timeout FALSE  
    LimitSwitches FALSE  
    UnexpectedChange FALSE  
}
```

5.6 Safety

This section describes the schema fields for the Safety sub-system.

5.6.1 Commands

SetSWESTop and SetSWEClose and SetSWESecure

Description:

Sets the software E-Stop, E-Close or E-Secure condition. In order to recover from this, the software E-Stop/Close/Secure needs to be cleared (ClearSWESTop/Close/Secure) and the E-Stop/Close/Secure state needs to be rest (ResetESTop/Close/Secure).

For details of the Emergency (E-Stop, E-Close and E-Secure) behaviours see Safety System above

Parameters: none

Returns: none

Example: DeviceServer.Devices.Safety.SetSWESTop {}

ClearSWESTop and ClearSWEClose and ClearSWESecure

Description:

Clears the E states as set above.

Parameters: none

Returns: none

Example: DeviceServer.Devices.Safety.ClearSWESecure {}

ResetEStop and ResetEClose and ResetESecure

Description:

All of the E- states are latched. So when all EE inputs (software, buttons, or other electrical) become inactive it is still required to send a ResetE* command to recover from the E state and allow normal operations.

Parameters: none

Returns: none

Example: DeviceServer.Devices.Safety.ResetEClose {}

ESecureHoldOff

Description:

When an E-Secure input becomes active it's associated countdown timer starts counting to zero. It is not until this timer reaches zero that the E-Secure state becomes active. An example of this is a UPS input to the Safety system. It is possible to configure the Safety node to accept this input and to not enter the E-Secure state until some amount of time after the input becomes active.

This command provides the higher-level software a chance to hold off on entering this E-Secure state by allowing it to reset all active countdown timers.

See also the ESecureHoldOffTime attribute.

Parameters: none

Returns: none

Example: DeviceServer.Devices.Safety.ESecureHoldOff {}

SetUPSHoldOffTime

Description:

This command sets the hold off time for the UPS input to the Safety system (as described above). It is expected that **all** IceStorm Series II domes will provide this functionality.

Depending on the dome's configuration there may be similar commands to control the hold off time for other input devices.

Parameters:

Parameters.Seconds

The number of seconds to hold off for. Due to the internal mechanisms the accuracy of this timer is ± 1 second.

DML Type: FLOAT

Units: seconds

Range: 16bit signed integer (0 – 32767)

Returns: none

Example: DeviceServer.Devices.Safety.SetUPSHoldOffTime { Parameters.Seconds 15.5 }

GetUPSHoldOffTime

Description:

Gets the hold off time (complete time, not current remaining) for the UPS input.

Parameters: none

Returns:

Returns.Seconds

DML Type: FLOAT

Units: seconds

Range: 16 bit signed integer (0 – 32767)

Example:

Send DeviceServer.Devices.Safety.Commands.GetUPSHoldOffTime {}

Receive DeviceServer.Devices.Safety.Commands.GetUPSHoldOffTime {

 Returns {

 Seconds 300.0

 }

}

5.6.2 Attributes

StopInputs

Description:

Shows the status of the current E-Stop inputs.

DML Type: STRUCTURE

SoftwareEStop

The current status of the software E-Stop input.

DML Type: BOOLEAN

EStop1Button through EStop6Button

The current status of the six E-Stop buttons positioned throughout the dome.

DML Type: BOOLEAN

Example:

```
DeviceServer.Devices.Safety.Attributes.StopInputs {  
    SoftwareEStop FALSE  
    EStop1Button FALSE  
    EStop1Button FALSE  
    EStop1Button FALSE  
    EStop1Button FALSE  
    EStop1Button FALSE  
    EStop1Button TRUE  
    EStop1Button FALSE  
}
```

CloseInputs

Description:

Shows the status of the E-Close inputs.

DML Type: STRUCTURE

SoftwareEClose

The software E-Close input.

DML Type: BOOLEAN

ECloseButton

The status of the single E-Close button (just inside the front door).

DML Type: BOOLEAN

Example:

```
DeviceServer.Devices.Safety.Attributes.CloseInputs {  
    SoftwareEClose FALSE  
    ECloseButton FALSE  
}
```

SecureInputs

Description:

Shows the status of the E-Close inputs.

Note that it is possible for an input to be active, and the count down timer not expired, meaning that the E-Secure state is not yet active.

It is possible that other inputs may be added to the dome (e.g. rain sensor etc.) and this would require an extension to the current schema.

DML Type: STRUCTURE**SoftwareESecure**

The status of the software E-Secure input. The software E-Secure input cannot be held off like the other electrical inputs.

DML Type: BOOLEAN

UPSInput

The status of the UPS input.

DML Type: BOOLEAN

Example:

```
DeviceServer.Devices.Safety.Attributes.SecureInputs {  
    SoftwareESecure FALSE  
    UPSInput FALSE  
}
```


EStopState and ECloseState and ESecureState

Description:

The status of the E states.

DML Type: BOOLEAN

Example: DeviceServer.Devices.Safety.Attributes.EStopState TRUE

ESecureHoldOffTime

Description:

The number of seconds until the E-Secure state becomes active (assuming the inputs remain active). If there are multiple active E-Secure inputs then this will be the input with the lowest current timer value, i.e. the one that will timeout first.

DML Type: INTEGER

Units: seconds

Example: DeviceServer.Devices.Safety.Attributes.ESecureHoldOffTime 285

5.7 Cooling

This section describes the schema fields for the Cooling sub-system. The cooling sub-system consists of three cooling systems named 'FanCoolingUnit1', 'FanCoolingUnit2', and 'FanCoolingUnit3'.

5.7.1 Commands

Enable

Description:

Turns on the cooling system. When cooling is on the system will attempt to maintain the temperature as set by the SetSetPoint command and as indicated by the CurrentTemperature attribute.

When the cooling system is on it will turn its fan on.

Parameters: none

Returns: none

Example: DeviceServer.Devices.FanCoolingUnit1.Commands.Enable {}

Disable

Description:

Turns off the cooling system. This will stop all temperature control and turn off the fan.

Parameters: none

Returns: none

Example: DeviceServer.Devices.FanCoolingUnit2.Commands.Disable {}

SetSetPoint

Description:

Set the desired temperature for the cooling system. The cooling system is responsible for obtaining the current temperature, which it generally does by averaging several sensors in its immediate vicinity.

FCU 1 is located in the equipment area and can generally be set and forgotten.

FCU's 2 and 3 are in the observing space and as a general rule will work together to achieve the set point. For this reason it is normal to set the set points for these two units either the same or very close together to avoid them 'fighting' each other. In most domes these two units share the input temperature sensors.

Parameters:

Parameters.Temperature

The required set point.

DML Type: FLOAT

Units: Degrees Celsius

Range: Any valid floating point number.

Returns: none

Example:

```
DeviceServer.Devices.FanCoolingUnit3.Commands.SetSetPoint {  
    Parameters.Temperature 2.5  
}
```

5.7.2 Attributes

Enabled

Description:

True if the unit is enabled and attempting to maintain temperature control.

DML Type: BOOLEAN

Example: DeviceServer.Devices.FanCoolingUnit1.Attributes.Enabled TRUE

CurrentTemperature

Description:

The current temperature that the unit is using as its input temperature.

DML Type: FLOAT

Units: degrees Celsius

Range: dependant on the types of sensors being used in the dome. Generally the range is adequate for the climate conditions expected at site.

Example: DeviceServer.Devices.FanCoolingUnit1.Attributes.CurrentTemperature 5.5

SetPointError

Description:

The difference between the set point and the current temperature (Setpoint – CurrentTemperature).

DML Type: FLOAT

Units: degrees Celsius

Example: DeviceServer.Devices.FanCoolingUnit2.Attributes.SetPointError 0.1

Output

Description:

The percentage output of the valve controlling the flow of coolant through the coil. This is mostly used for debugging and tuning.

DML Type: FLOAT

Example: DeviceServer.Devices.FanCoolingUnit1.Attributes.Output 50.0

5.8 Lighting Circuit

The lighting systems on domes tend to differ in their capabilities and control required.

The commands and attributes below are those that are considered 'normal' or ideal. The interface for different enclosures may be required to change with differing lighting controls.

5.8.1 Commands

TurnOn

Description:

Will instruct the lighting controller to turn on the lights. Depending on the wiring of the system this may turn on all the lights, some predetermined sub set, or not turn on any lights.

Parameters: none

Returns: none

Example: DeviceServer.Devices.HouseLights.Commands.TurnOn {}

TurnOff

Description:

This command is also dependant on the wiring of the specific dome, though it will generally turn off all the lights in the enclosure.

Parameters: none

Returns: none

Example: DeviceServer.Devices.HouseLights.Commands.TurnOn {}

5.8.2 Attributes

AllOff**Description:**

Again, each enclosure has a slightly different wiring configuration when it comes to lighting controllers.

This attribute is only true if the system can determine that all the lights connected to it are currently off.

DML Type: BOOLEAN

Example: DeviceServer.Devices.HouseLights.Attributes.AllOff FALSE

5.9 Recirculation Fans and the Snow Melt Cable, (Other Digital I/O Channels)

This section describes the interfaces to the Recirculation Fans (RecirculationFan1 and RecirculationFan2) and the Snow melt cable (SnowMelt). These are all basic digital Input/Output controls.

5.9.1 Commands

Enable

Description:

Activates the digital output. In the case of the fans this will start the addressed fan spinning. In the case of the snow melt, this will start it warming up and melting any ice that has formed in the cable troughs.

Parameters: none

Returns: none

Example: DeviceServer.Devices.SnowMelt.Commands.Enable {}

Disable

Description:

De-activates the digital output. This will stop the fans, and turn off the snow melt cable heating.

Parameters: none

Returns: none

Example: DeviceServer.Devices.RecirculationFan1.Commands.Disable {}

5.9.2 Attributes

OutputEnabled

Description:

Will be true is the digital output is enabled.

This status will generally come through an actual feedback input rather than echo the output status, meaning that it will generally be possible to see if the connected device is not functioning. In the case of the snowmelt the two individual cables are wired in series, giving a logical AND of both inputs (so that the input is only true if **both** snowmelt inputs are active).

As with the other 'changeable' components in an enclosure check with your specific wiring diagrams to determine the actual functionality of your facility.

DML Type: BOOELAN

Example: DeviceServer.Devices.SnowMelt.Attributes.OutputEnabled TRUE