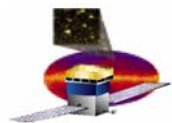


# GLAST Large Area Telescope: I&T Integration Readiness Review

Online Peer Review  
July 21, 2004

Core I

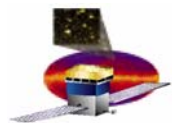
Selim Tuvi  
I&T Online  
SLAC



## LATTE Overview

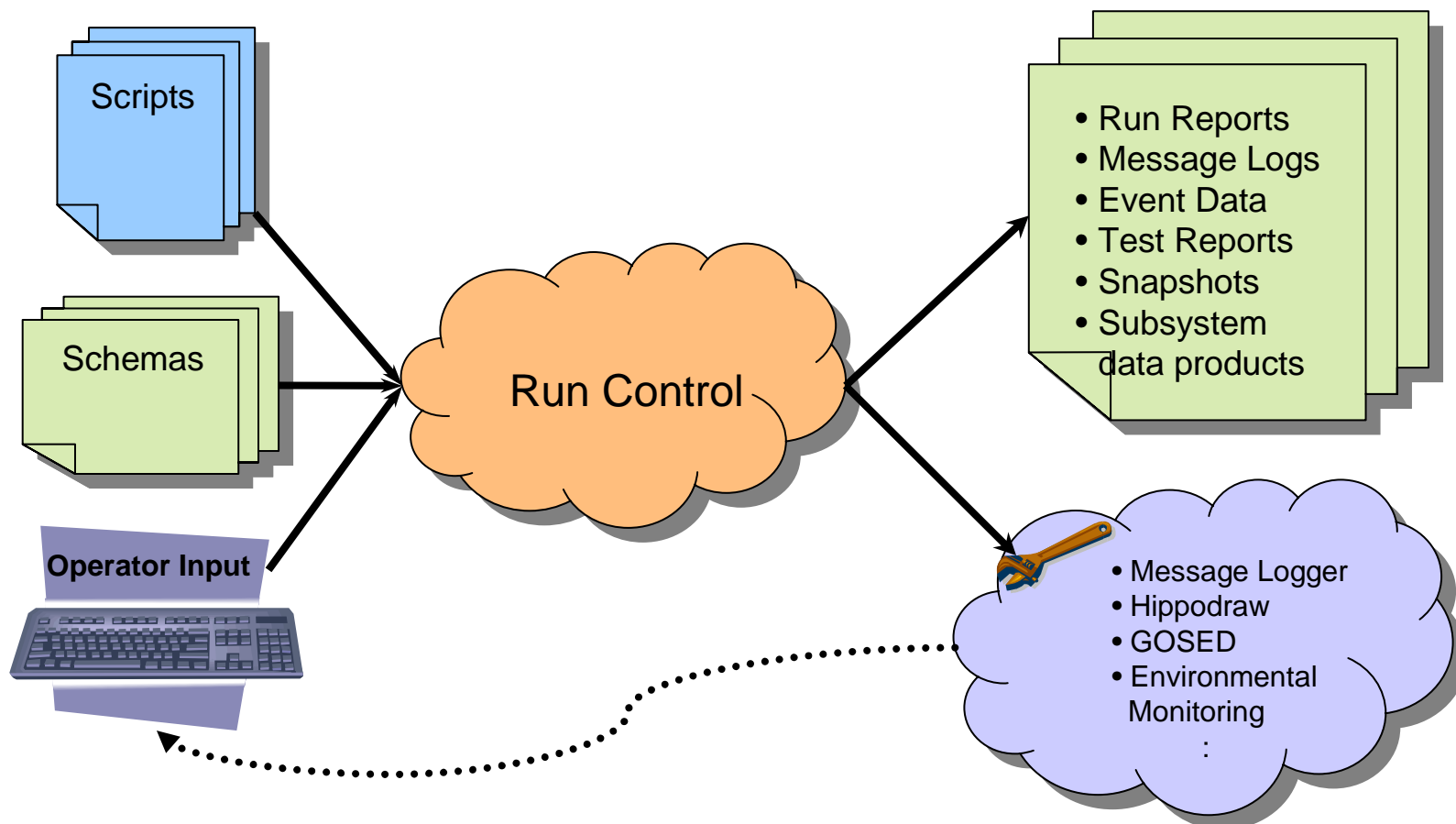
---

- **Test Executive Architecture**
  - Run Control
  - Schema and Configuration
  - Servers (Command and Event)
  - Security Framework
- **Software Distribution**
  - Versioning
  - Making a Release
  - Release Verification



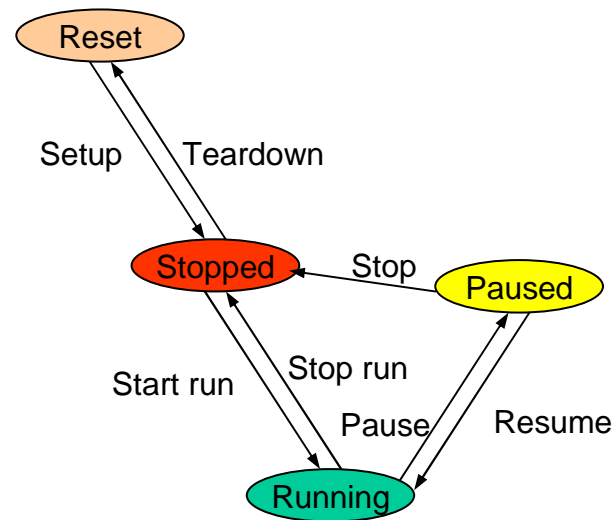
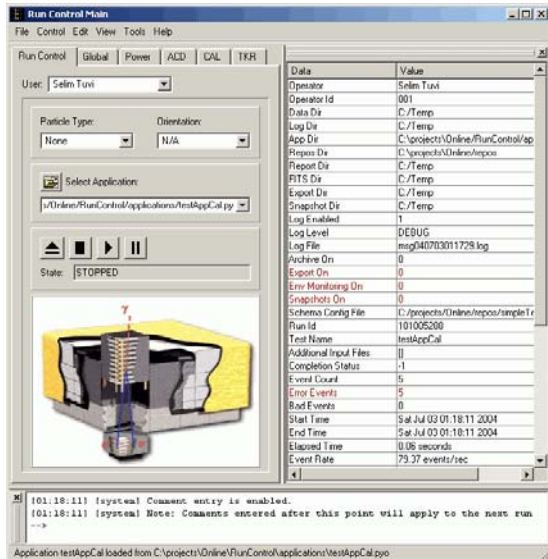
## Run Control - Overview

- A framework and user interface for subsystems to conduct testing and data collection.

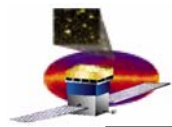


- Scripts and schemas are certified by Online prior to release to I&T.
- Validation on operator specified input parameters required.
- Feeds data into various tools that are available to the operator

# Run Control - Main Features



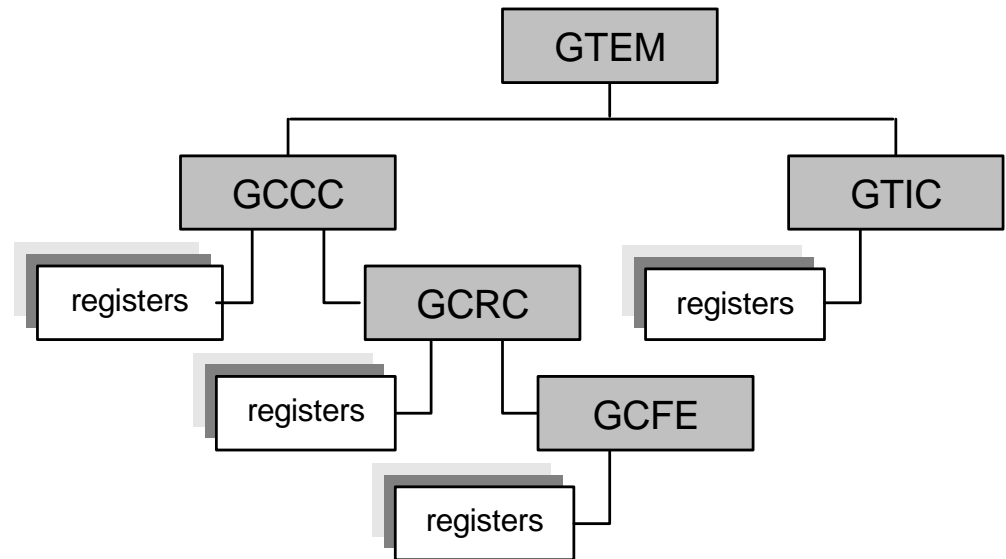
- **Main Features**
  - A scripting environment driven by a state machine.
  - Message Logger
  - Environmental Monitoring GUI
  - Test Reports
  - Test Suites
  - E-logbook integration (including Run comments)
  - Hippodraw access (data visualization)
  - Event Data Distributor
  - Python Shell access (debugging support)

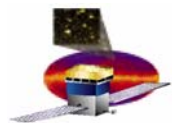


## Schema and Configuration - Schema

- Provides a hierarchical representation and interface to the underlying hardware.
  - XML based.
  - Can be parsed by tools written outside I&T/Online.
  - Common with FSW.
  - Schema mirrors hardware:

```
<?xml version='1.0' encoding='UTF-8'?>
<LATdoc name='CAL Schema'>
  <schema>
    <GLAT>
      <GTEM ID='0'>
        <GCCC ID='0-3'>
          <GCRC ID='0-3'>
            <GCFE ID='0-11' />
          </GCRC>
        </GCCC>
      </GTEM>
    </GLAT>
  </schema>
</LATdoc>
```

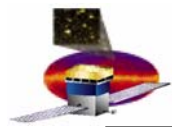




## Schema and Configuration - Configuration








- **Configuration**
  - Schema defines the hardware.
  - Configuration sets up the hardware.
  - Multiple configurations can be defined per schema.
  - Configurations are selectable and can be reapplied.
  - Example:

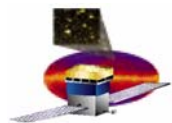
```
<?xml version='1.0' encoding='UTF-8'?>
<configuration name='testConfig' version='1.0'>
  <GLAT>
    <GTEM ID='0'>
      <GTCC ID='5'>
        <GTRC ID='0'>
          <GTFE ID='11'>
            <data_mask>0x20</data_mask>
            <calib_mask>0x20</calib_mask>
            <trig_mask>0x20</trig_mask>
            <mode>0x1</mode>
          </GTFE>
        </GTRC>
      </GTCC>
    </GTEM>
  </GLAT>
</configuration>
```



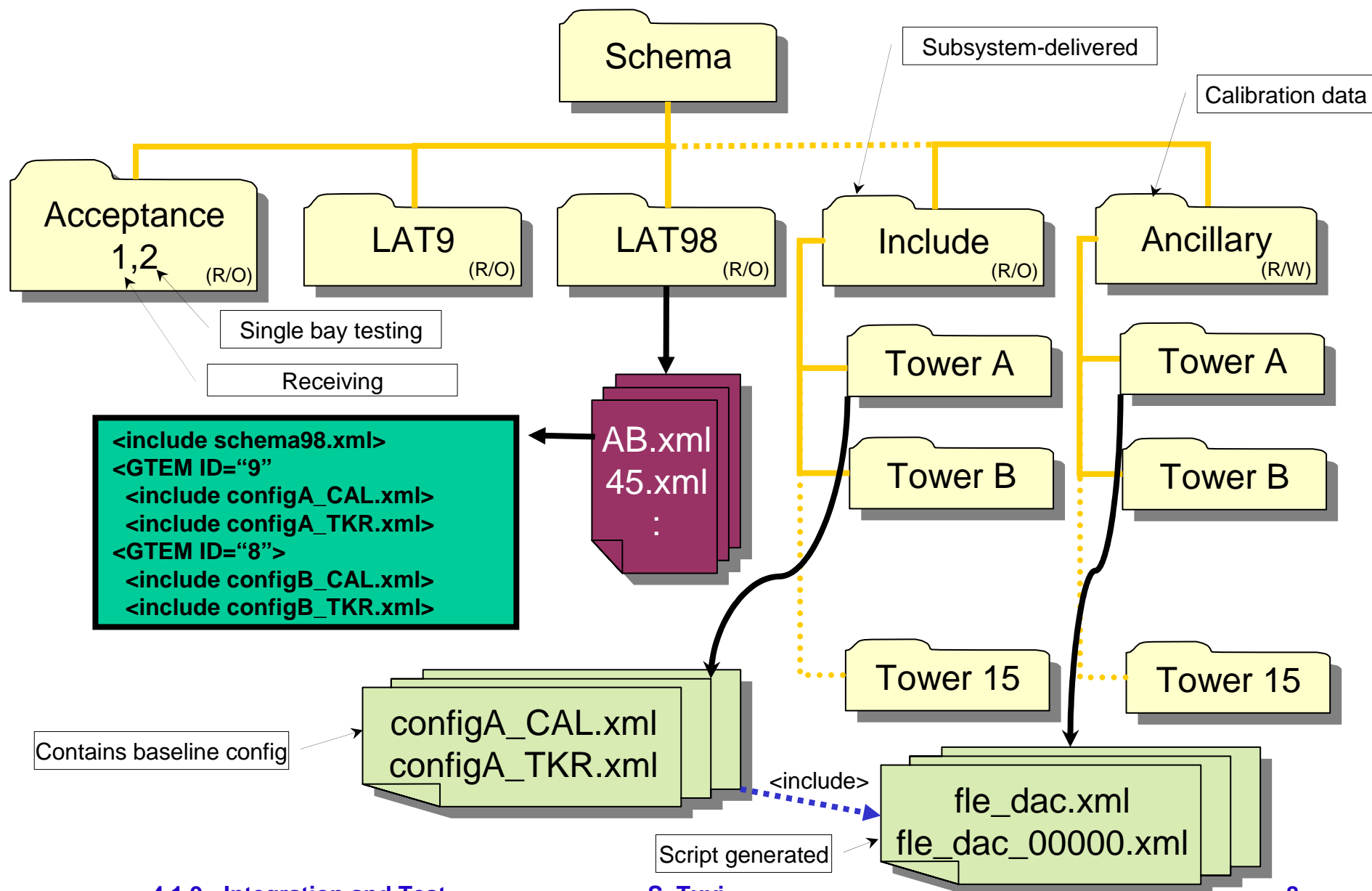
## Schema and Configuration - Other Features

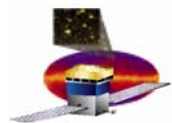
---

- **Other Features**
  - **EGUs (Engineering unit conversion)** 
  - **Constraints (prevents writing of undesirable values)** 
  - **Rules (allows limit checking and alarming)** 
  - **<include> (allows gluing together of fragments)** 
  - **<opaque> (application defined)** 
  - **<badStrips>** 
  - **LATc (configuration blobs)** 



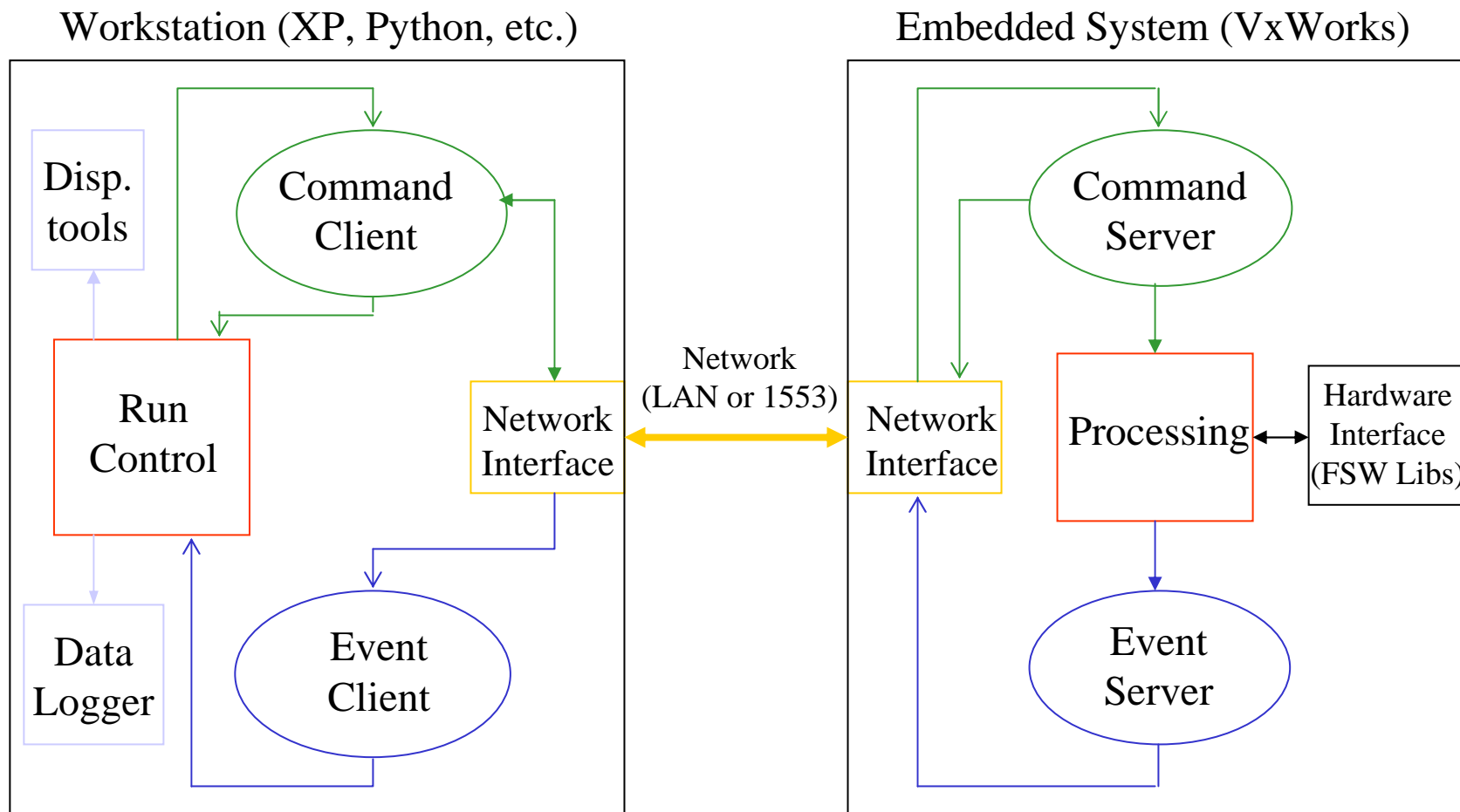
# Maintaining Configuration Control During I&T Testing

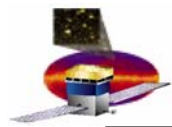




## Command and Event Server - Overview

- Teststand software block diagram



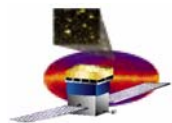


## Command and Event Server - Overview (cont.)

---



- Simple syntax for accessing the hardware registers.
  - Example:

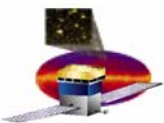
```
self.lat.TEM[0].CONFIGURATION = 0x80000000L
```
- In addition to providing access to the registers, Command Server features the following:
  - Deadtime measurement.
  - Timestamping of commands.
  - Various statistics.
  - VME bus memory access.
  - Access to various support functions from LATTE.
- Event client and Event Server
  - FSW delivers events to the event server for transport to the event client.
  - Prescaling and filtering of events.
  - Provides methods for reading and parsing event data.
  - Event data handling support for scripts through RunControl



## Security - Overview

---

- **Main Features** 
  - Run Control command-line option.
  - User authentication.
  - Permissions to provide feature restriction.
  - Roles: groups of permissions.
  - Subsystem scripts can add their own permissions by using the security API. 
  - Kiosk mode (limits certain Windows features).



## Security – Script Input Verification

- Input needs to be signed off on using the operator's password.
- A work in progress.

Parameter	Value	
Layers Selection	{X0,Y0}	<input checked="" type="checkbox"/>
No. of strobes per setting	1	<input checked="" type="checkbox"/>
Thr DAC	30	<input checked="" type="checkbox"/>
Thr Range	0	<input type="checkbox"/>
Cal DAC	20	<input type="checkbox"/>
Cal Range	1	<input type="checkbox"/>

**Selim Tuvi**, by checking the parameter-verified boxes above, providing your password below and clicking on 'Confirm', you certify that all parameters listed are set correctly for the **testAppCal** test about to be performed. If you disagree with one or more of the parameter settings, click on 'Cancel' below and the system will be brought back to the STOPPED state.

Password:

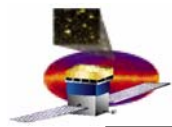
### Selim Tuvi:

- by checking the parameter-verified boxes above,
- providing your password below and
- clicking on 'Confirm'

You certify that:

- Parameters listed are set correctly for the **testAppCal** test about to be performed.

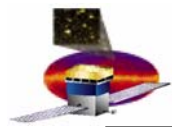
If you disagree with one or more of the parameter settings, click on 'Cancel' below and the system will be brought back to the STOPPED state.



## Versioning

---

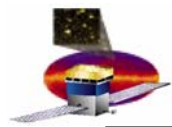
- Configuration management plan is given in document LAT-MD-03492.
- Components under version control (CVS):
  - Core LATTE code, example test scripts and schemas.
  - Subsystem scripts and schemas released to I&T.
  - Flight model test configuration files.
- Before every release, the modules are tagged with the release number



## Making a Release – LATTE Releases

---

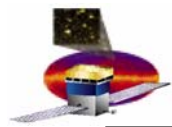
- **LATTE Releases**
  - Schedule based on FSW releases, subsystem needs and CCB mandates.
  - Online resolves subsystem issues using the issue tracking system.
  - Online receives the hardware interface libraries from FSW.
  - A release candidate is produced and tested against the hardware.
  - The release is tagged as Pxx-yy-zz and an installer script creates the Windows executable package.
    - xx – Changes that are non-backward compatible
    - yy – New features and changes.
    - zz – Bug fixes
  - Online web site is updated with release notes and various documentation.
  - The release is announced to the distribution list by e-mail.



## Making a Release – Subsystem Releases

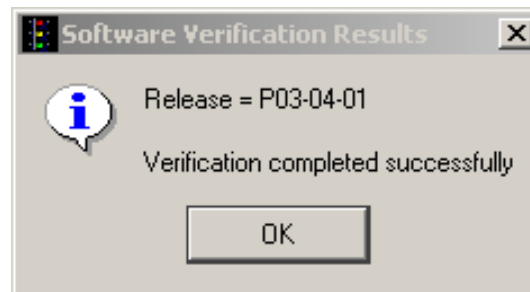
---

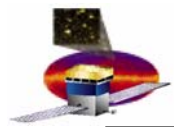
- **Subsystem Releases to I&T by Online**
  - Subsystems use the same CVS repository to maintain their release.
  - Subsystems tag their release in CVS
  - Online retrieves the subsystem tag and goes through the validation and certification process.
  - Online releases the subsystem release to I&T
  - Subsystems that provide scripts to I&T:
    - CAL
    - TKR
    - ACD
    - ELX
    - E2E



## Release Verification

- Online maintains a checksum database for LATTE and for each subsystem release.
- As tests are conducted, the modules used are checked against these databases.
- Modules and XML files which fail the verification are logged to the run report.
- Verification is also available as a menu item in Run Control:

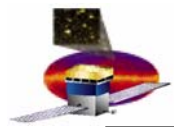




## Questions

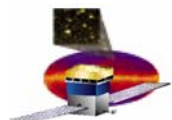
---

**Questions?**



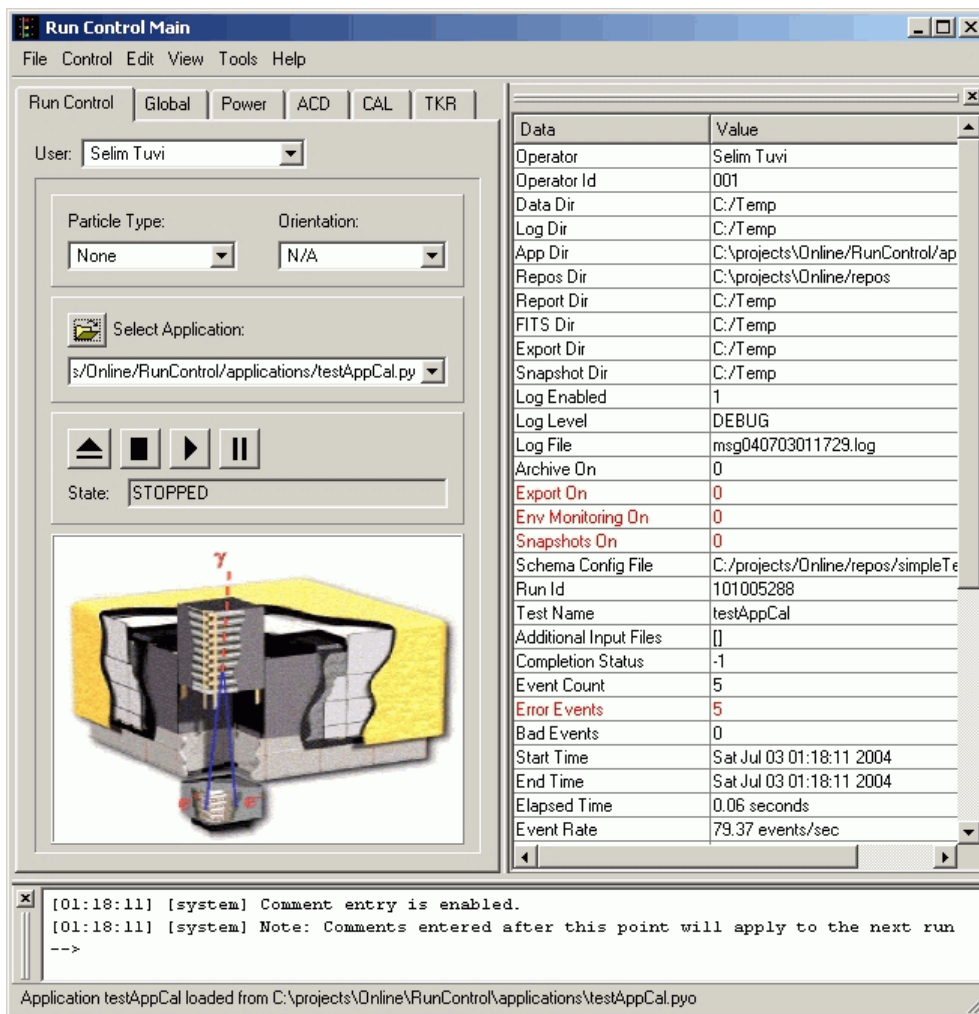
# Backup Slides

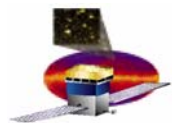
---



# RunControl – Main GUI

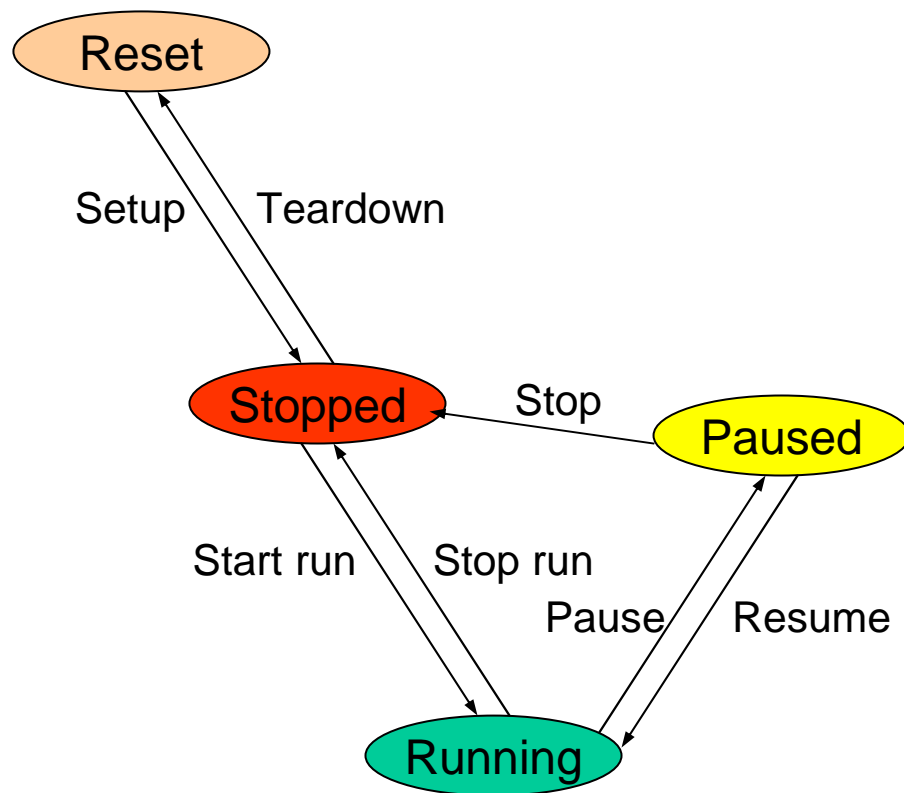
- Main RunControl GUI

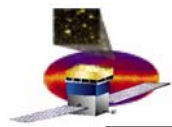




## RunControl – State Machine

- State Diagram

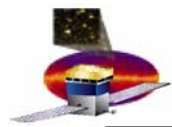




## RunControl - Operation

---

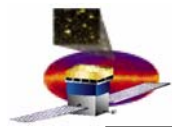
- **Sample procedure for an operator using RunControl**
  - Turn on the VxWorks crate and wait for the startup script to execute.
  - Launch RunControl from the desktop icon and login.
  - Load a schema.
  - Power up the front ends.
  - If necessary select the particle type and orientation.
  - Load a script.
  - Load a configuration (optionally select a new schema)
  - Enter the input parameters and sign off on them.
  - Execute the script.
  - Enter comments during or at the end of execution.



## Run Control – Command Line Switches

---

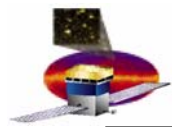
- **Command Line Switches:**
  - c or --config to specify the run control configuration file to use (defaults to runControl.cfg in the current directory).
  - d or --debug to specify event client debug mode
  - D or --cmddebug to specify command client debug mode
  - h or --help for this help.
  - p or --playback to indicate that Run Control should command the event server to play back the next event from its file.
  - n or --noreload to specify a file that contains prefixes of modules which should not be reloaded when a user script is selected.
  - s The default schema file to load. If this switch is specified then the operator does not get asked for a schema during setup.
  - S The full path to the secure directory where the password file ("passwords") and the user permissions file ("security.cfg") is stored.
  - u or --userid to specify the User Id.



## Run Control - State Transitions

---

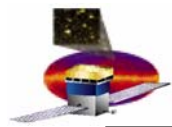
- **Setup**
  - Schema is loaded.
  - Configuration (if any) is applied to the hardware.
  - Script's setup method is executed.
  - Trigger is setup.
- **Start Run**
  - Script's startRun method is executed.
  - If enabled, a snapshot of the hardware based on the input schema is generated.
  - A sweep event is solicited (to sweep out buffered events).
  - Triggers are enabled.
- **Stop Run**
  - Triggers are disabled.
  - A sweep event is solicited (to sweep out buffered events).
  - Script's stopRun method is executed.
  - A run report is generated.
  - If enabled, a snapshot is generated.
  - All input and output files are copied to an export directory.



## Run Control - Other Features

---

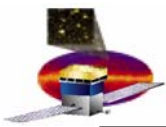
- **Hippodraw access**
  - Run Control internally creates an instance of a Hippodraw canvas and exposes it to the user scripts.
- **Event Data Distributor**
  - Run Control can optionally broadcast event data to external consumer tools such as GOSED (GLAST Online Single Event Display).
- **Message Logger**
  - Run Control uses the Python logging package.
  - Log messages can come from Run Control itself or from scripts.
  - All messages are sent to a file that exists for that session.
  - They are also sent to the Message Logger GUI.
- **Environmental Monitoring GUI**
  - During a Run Control session operators can monitor environmental quantities using this GUI.
- **Python Shell access**
  - For administrators, access to a Python shell from within Run Control is invaluable.
- **Test Reports**
  - User scripts can create reports in HTML that contain test results.



## Run Control - Other Features (cont.)

---

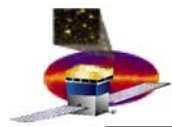
- **E-logbook integration**
  - Run reports can be recorded in real-time in a local MySQL database and accessed by the standalone Elogbook.
- **Test Suites**
  - Scripts can be set up to run in an unattended fashion by writing a test suite script that coordinates its subscripts execution. Multiple layers (recursion) can be employed.



## Other Schema/Configuration Features (cont.)

- **EGUs (Engineering unit conversion classes)**
  - Allows conversion from raw units to engineering units.
  - Can be defined under a <schema> tag or a <configuration> tag.
  - Subsystem defined EGU classes can be incorporated using the <import> tag in the <declarations> section.
  - **Example:**

```
<egu name="TEM 3.3 Voltage">  
  <class>gEGU.GEGU_linear</class>  
  <parameters>  
    <slope>0.001221</slope>  
    <intercept>0.0</intercept>  
    <units>V</units>  
  </parameters>  
</egu>  
<GLAT>  
  <GTEM ID='0-15'>  
    <GTIC>  
      <adc_tem_digital_3_3v egu="TEM 3.3 Voltage"/>  
    </GTIC>  
  </GTEM>  
</GLAT>
```

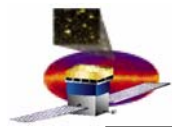


## Other Schema/Configuration Features (cont.)

- **Constraints**

- Provides a mechanism to limit the values that can be set on a register.
- If a value is rejected through the constraint, an exception is raised.
- **Example:**

```
<constraint name="con_veto_delay">
  <class>userCon.Constraint2</class>
  <parameters>
    <name>limit</name>
    <subsystem>acd_group</subsystem>
    <category>veto_group</category>
    <enabled>TRUE</enabled>
    <continuous>TRUE</continuous>
    <low>150</low>
    <high>1700</high>
  </parameters>
</constraint>
<GLAT>
  <GAEM>
    <GARC ID="0 - 11">
      <veto_delay egu="egu_veto_delay" constraint="con_veto_delay"/>
    </GARC>
  </GAEM>
</GLAT>
```

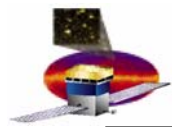


## Other Schema/Configuration Features (cont.)

- **Rules**

- Defines the action to be taken when the register value is read (evaluated).
- Can be used to make alarms go off.
- **Example:**

```
<rule name='voltageMonitor'>
  <class>gRule.GrRuleAlarms</class>
  <parameters>
    <name>voltageMonitor</name>
    <subsystem>CAL</subsystem>
    <category>voltage</category>
    <enabled>1</enabled>
    <continuous>1</continuous>
    <yellowLowerLimit>3.3</yellowLowerLimit>
    <yellowUpperLimit>3.31</yellowUpperLimit>
    <redLowerLimit>3.25</redLowerLimit>
    <redUpperLimit>3.4</redUpperLimit>
    <actions>
      <yellowAction/>
      <redAction/>
    </actions>
  </parameters>
</rule>
```



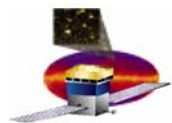
## Other Schema/Configuration Features

- **<include> tag**
  - Provided to facilitate schema reuse.
  - Schema fragments can be shared among many schemas.
  - Example:

```
<schema>
  <GLAT>
    <GTEM ID='0'>
      <include filename="$ONLINE_ROOT/repos/TKRTowerInfo.xml"/>
    </GTEM>
  </GLAT>
</schema>
```

- **<opaque> tag**
  - Subsystem defined XML fragment.
  - Can contain arbitrary content.
  - Subsystem's responsibility to parse the data.
  - Example:

```
<?xml version='1.0' encoding='UTF-8'?>
<opaque name="TKRtowerInfo">
  <type>Minitower</type>
  <towerSerial>xxx-xxx-xxx</towerSerial>
  <layer id="X1">
    <mcm>102</mcm>
  </layer>
</opaque>
```



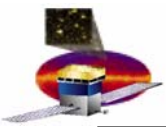
## Other Schema/Configuration Features (cont.)

- **<badStrips> tag**
  - Used to specify the dead and noisy strips for Tracker EM tests.
  - Uses DTD from Offline.
  - Example:

```
<badStrips badType= "hot">
  <generic instrument="EM" timestamp="2003-2-5-15:58" calType="hotStrips" fmtVersion="v2r0" >
    <inputSample startTime="0" stopTime="0" triggers="physics" mode="normal" source="stuff" >
      Output from LSR bad strips alg, running on MC data
    </inputSample>
  </generic>
  <tower row="0" col="0">
    <!-- (6,7,0) X1 -->
    <uniplane tray="3" which="bot" howBad="1" >
      <stripList strips= " 768-1532
                          " />

    </uniplane>
    <!-- (4,5,0) Y1 -->
    <uniplane tray="2" which="bot" howBad="1" >
      <stripList strips= " 122
                          330
                          1472
                          1528-1529
                          1533-1535
                          " />

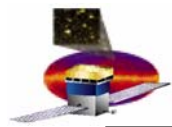
    </uniplane>
  </tower>
</badStrips>
```



## Schema/Configuration and FSW

---

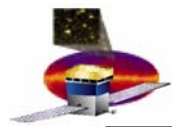
- **LATc**
  - Interface provided by FSW.
  - Will replace snapshots in LATTE.
  - Triggering the creation of configuration blobs.
  - Receiving configuration blobs.



## Schema/Configuration To Do

---

- **Optimize schema loading for the full LAT.**
- **Allow tailoring of the loaded schema with a user interface.**
- **Include a list of environment variables used by the schema and its include files.**
- **Provide verifyConfig, writeConfig methods**
- **Integrate LATTE with the FSW LATc package.**
- **Check the STATUS reg of the parent component after every SET in the configuration.**
- **Implement the <split> tag in the configuration for TKR.**
- **Generic node tags.**

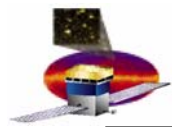


## OCS & OES – Overview (cont.)

---

- LATTE has a simple syntax for accessing the hardware registers.
  - **Example:**  

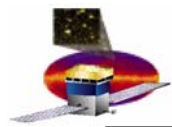
```
self.lat.TEM[0].CONFIGURATION = 0x80000000L
```
- LATTE communicates with the hardware over a TCP/IP connection.
- LATTE's CmdCli (command client) class translates the request to a mnemonic and packages it into a TCP/IP packet.
- OCS (Online Command Server) which runs on the VxWorks SBC, listens for these packets, decodes them and calls the appropriate Flight Software library functions.
- The result gets packaged in a similar fashion and gets sent back to LATTE over the same TCP/IP connection.
- CmdCli interprets these packages, decodes them and sends the result back to the caller.



## OCS & OES - Overview (cont.)

---

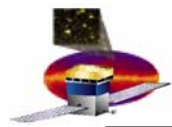
- **EvtCli and OES (continued)**
  - **LATTE integrates event data handling seamlessly in RunControl:**
    - Every time an event is received, RunControl calls the currently executing script's process method with the event data buffer.
    - In process(), the user script can call EvtCli's parsing methods to access the different components of the event.
  - **EvtCli and OES also provides the following support to LATTE:**
    - Prescaling events.
    - Support for ancillary data provided by subsystems.



## OCS & OES - FSW

---

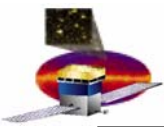
- **OCS and OES is part of the FSW release and considered part of the FSW packages.**
  - **Currently maintained by I&T**
  - **Build and release mechanism same as other FSW packages.**
  - **When LATTE is about to be released, a “tarball” containing FSW libraries including OCS & OES is requested. The tarball contents gets CVSd and included in the LATTE release.**



## Security - Overview

---

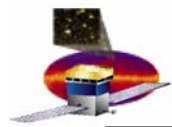
- **RunControl command-line option.**
  - **-S=path\_to\_password\_and\_permissions\_file**
- **User authentication.**
  - **Based on a login id and password.**
  - **Passwords are encrypted using base64 and sha.**
- **Permissions to provide feature restriction.**
  - **Simple allow or disallow type scheme.**
- **Roles: groups of permissions.**
  - **Each role can contain a number of permissions**
- **Kiosk mode.**
  - **Set using the policy manager under Windows.**
  - **Disables certain features of the GUI.**
  - **As soon as the operator logs in, RunControl is launched.**
  - **Required for the security framework to operate in a secure way (i.e. Non kiosk environments can still be “hacked”).**



## Security - Registering Custom Permissions

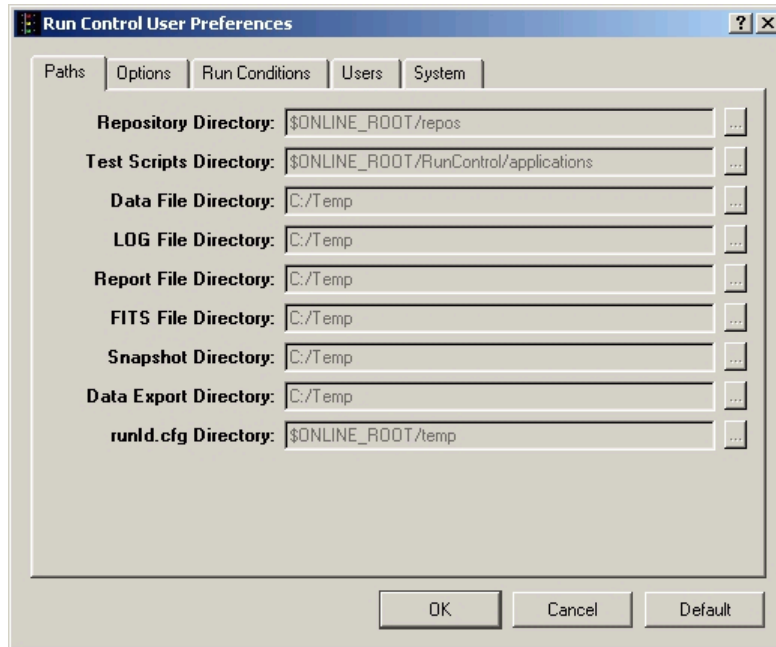
---

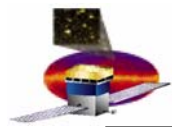
- Subsystem scripts can add their own permissions by using the security API.
- Example:
  - `self.registerPermission("tkr_operator", "select_layers", "Allow the selection of layers")`
  - `self.checkPermission(operator, "select_layers")`
  - The role "tkr\_operator" needs to exist and have users belong to it prior to script execution.



## Security – Limiting Access

- Preferences screen for the Operator showing disabled fields





## Security – User Maintenance

- User Maintenance Screen

The dialog box 'Run Control User Preferences' has tabs for Paths, Options, Run Conditions, Users, and System. The 'Users' tab is active, showing a list of users and fields for adding or modifying a user.

**User ID:** 001

**User Name:** Selim Tuvi

**Login Id:** stuvi

**Password:** [masked]

**Password (again):** [masked]

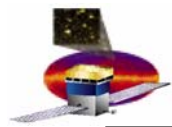
**Old Password:** [empty]

**Buttons:** Save User, Change Password, Delete User

**Users List:**

ID	Name	Login Id
001	Selim Tuvi	stuvi
002	Ric Claus	claus
003	Jim Panetta	panetta
004	Eduardo do Couto e Silva	Eduardo
005	Larry Wai	lwai
006	Elliott Bloom	Elliott Blc
007	Hiro Tajima	Hiro Taji
008	Xin Chen	Xin Cher
009	Tsunefumi Mizuno	Tsunefu
010	Tune Kamae	Tune Ka
011	Brian Grist	Brian Gri
012	Gary Godfrey	Gary Go
013	Berrie Giebels	Berrie Gi
014	Brian Horwitz	Brian Ho
015	John Canfield	John Can

**Buttons:** OK, Cancel, Default



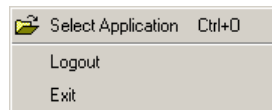
## Security - User Login/Logout

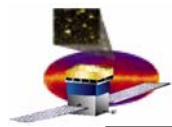
---

- **Run Control Login Dialog**



- **Logout menu item**





## Security – Sample Permissions File

---

### [users]

# If a user is not listed then he/she has no permissions  
# The administrator role is a special role which automatically  
# grants the user all the permissions.  
# There may be more predefined roles which contain permissions  
# based on a category.

# userid = role1, role2, role3, ...  
stuvi=administrator  
claus=operator

### [roles]

# role\_name = perm\_name1, perm\_name2, ...

operator=set\_particle\_type, set\_orientation, enable\_power\_panel, enable\_power\_up,  
edit\_preferences, edit\_paths, edit\_options, edit\_users, edit\_system,  
set\_visual\_system\_pref

### [permissions]

# permission\_name = Permission Description  
allow\_python\_shell=Allow access to the Python shell  
set\_particle\_type=Allow setting of the particle type  
set\_orientation=Allow setting of the orientation  
enable\_global\_panel=Allow access to the Global panel  
....