Technical Report:

Automated FACSUM Data Acquisition Tool for the Large Binocular Telescope (LBT)

Software Version: Bash Script (FACSUM Downloader)
Author: Felice Cusano, Andrea Rossi, Affiliation: INAF – Istituto Nazionale di Astrofisica – Osservatorio di Astrofisica e Scienza dello Spazio Bologna (OAS), via Gobetti 93/3 40129 Bologna - ITALY
Date: 14 Feb 2026

---

1. Introduction: The Large Binocular Telescope (LBT)

The Large Binocular Telescope (LBT), located on Mount Graham in Arizona, operates under a complex system of environmental monitoring and telemetry acquisition. Continuous access to real-time observatory parameters is essential for both operational awareness and post-observational analysis.

This report describes a lightweight Bash-based software tool designed to automatically retrieve environmental and system status data from the LBT FACSUM service.

---

2. Software Purpose

The software performs automated acquisition of:

- FACSUM parameter logs, containing observatory telemetry
- All-sky images, providing visual sky conditions

The tool is intended to:

- build a local archive of observatory conditions
- support monitoring and diagnostics
- provide input data for subsequent analysis tools

---

3. Data Sources

The script retrieves data from the LBT data management system (DMS):

- FACSUM parameters:

https://dms.mountain.lbto.org/webasm/facsum/data/params.txt

- All-sky image:

https://dms.mountain.lbto.org/webasm/facsum/data/AllSkyCurrentImage.JPG

These sources provide real-time observatory status information.

---

4. Software Architecture

4.1 Timestamp Generation

A UTC timestamp is generated at runtime:

TS=$(date -u +%Y%m%d_%H%M)

This timestamp is used to uniquely identify all downloaded data products.

4.2 FACSUM Log Download

The script retrieves the parameter file using `curl`:

- silent mode (`-s`)
- follow redirects (`-L`)

The output is stored locally as:

facsum_<timestamp>.txt

This creates a chronological archive of observatory telemetry.

4.3 All-Sky Image Download

The all-sky image is downloaded using the same timestamp to avoid caching issues:

curl -sL "${ALLSKY_URL}?t=${TS}"

The timestamp is appended as a query parameter, ensuring that the most recent image is retrieved.

The image is stored as:

allsky_<timestamp>.JPG

4.4 Output Directory

All downloaded files are stored in:

$/home/facsum_logs/

This directory acts as a local repository for both numerical and visual monitoring data.

4.5 Error Handling

The script uses:

set -euo pipefail

This ensures:

- immediate exit on errors
- detection of undefined variables
- proper handling of pipeline failures

This design increases robustness and prevents silent failures during automated execution.

---

5. Output Description

At each execution, the software produces:

- one FACSUM log file (`facsum_<timestamp>.txt`)
- one all-sky image (`allsky_<timestamp>.JPG`)

These files can be subsequently processed by analysis tools, such as the environmental monitoring and visibility analysis software described in related reports.

---

6. Operational Use

The script is designed to be executed periodically (e.g., via `cron`) to create a time series of observatory conditions.

Typical use cases include:

- real-time monitoring

- nightly logging
- long-term environmental analysis
- support for observing reports

---

7. Integration in Workflow

This tool represents the data acquisition layer of a broader workflow:

1. Data acquisition (this script)
2. Data extraction and formatting (FACSUM parser)
3. Scientific/operational analysis

This modular design ensures flexibility and reusability.

---

8. Conclusions

The developed software provides a simple and reliable method for acquiring real-time observatory data from the LBT FACSUM system.

Its main advantages include:

- minimal dependencies
- automated data retrieval
- consistent timestamp-based archiving
- compatibility with downstream analysis tools

The tool is suitable for continuous monitoring and integration into automated pipelines.

The software described in this report is available from the authors upon request.

Future improvements may include:

- error logging and retry mechanisms
- configurable output paths
- integration with databases or monitoring dashboards