

# Outline of Fault Tolerance Document

A new version of this document is attached at the end. (At least) one more editing pass is still slated to occur.

## Fault Tolerance Goals [KTL; RP, GPDF]

*This section is now expanded at [FaultToleranceGoals](#).*

Key fault tolerance goals for the overall DMS:

- Preserve raw images, calibration data, metadata, and associations between them
- Ensure system is running as intended in order to meet the requirements for availability of data
  - ◆ Meet alert publication requirements
  - ◆ Meet nightly archive processing requirements
  - ◆ Meet data release processing requirements
  - ◆ Meet catalog query requirements
  - ◆ Meet ad hoc science pipeline processing requirements
- Achieve maximum fault tolerance (for the above goals) within existing budget limits
  - ◆ Equipment
  - ◆ Development
  - ◆ Operations

Need to be able to address failures with a system that is within budget for materials and personnel. Automation will be required to keep personnel budget within limits. Level of detail of planning will be less for low-probability events and those that have large resource requirements for recovery.

To meet the above goals, a combination of:

- Hardware infrastructure
- Middleware
- Application software
- SDQA

will be needed.

## Requirements [GPDF; RRL]

Requirements derive from:

- [DM Functional Requirements](#)
- [Science Requirements](#)

## Interfaces [RP; KTL]

[\[FaultToleranceInterfaces\]](#)

SDQA may trigger alerts to humans: not fault tolerance

# Faults and Use Cases [RRL; JB]

*This section is now expanded at [FaultToleranceDocumentUseCases](#).*

What are faults?

- How probable is each fault?
  - ◆ Human error will be probable in the beginning
  - ◆ Will improve as we can detect over time
  - ◆ Can we prevent human error up front? Automation, best practices can help
  - ◆ Are human errors different from other failures of the same component?
    - ◇ May be more likely to be detected by QA than other systems

Working notes: Basic properties of faults, list of faults, use cases, etc.

## Fault Tolerant Design Patterns

### Detection [GD; SL]

*This section is expanded at [FaultDetection](#).*

- Pre-testing
- Checksums
  - ◆ Immediate read-back verification (image level, whole tape level)
  - ◆ Delayed "scrubbing" verification
- Watchdog/heartbeat
- Self-reporting

### Designs [JB; SL]

*This section is now expanded at [FaultToleranceDocumentDesigns](#).*

This is a list of fault tolerant design strategies for components of the DMS.

- Redundancy/failover
- On-time partial or degraded delivery with no delivery of failed data
- On-time partial or degraded delivery with spare capacity used to deliver failed data later
  - ◆ Software reloading from (e.g. previous day's) backup version
  - ◆ Data reloading from alternate source
  - ◆ Checkpointing to reduce delivery delays

Dynamic system reconfiguration is a requirement for all of the above.

These are design practices that application developers need to be aware of when choosing algorithms and implementing them:

- Limiting/eliminating overwrites
- Minimizing failure impact; propagating failures to other affected processing
  - ◆ Minimize data dependencies across parallel processes
- Segregating mutable and immutable data
- Middleware will treat uncaught application exceptions as permanent failures
  - ◆ Resource exhaustion faults may be retried (ideally in less-constrained environment)
- No I/O (or database queries) by application stages
  - ◆ Allows middleware to capture faults and retry or recover
- Mark all degraded results in appropriate metadata

## Software Frameworks [GD; SL]

- Pipeline framework
- Failover framework (DB, message broker)

## Development Plan Impact [KTL, RP; RRL]

*This section is now expanded at [FaultToleranceImpacts](#).*

Describe the high-level design patterns used for key parts of each of the following systems:

- Infrastructure systems such as routing, DNS, authentication, etc.
- Raw data ingest
  - ◆ Interface document is necessary:
    - ◇ Checksum as early as possible
    - ◇ Acknowledgment protocol for informing when data has been persisted safely (2 copies always)
    - ◇ Delivery of data to processing systems
  - ◆ Distribution of data replica(s) to DACs (based on resources/external factors, metadata, data?)
- Alert pipeline processing (many subsystems)
- Nightly archive processing (many subsystems)
- Image storage and retrieval
  - ◆ Tape scrubbing, including replica at DACs (DR processing, technology migration)
- Data release processing (catalog generation) (many subsystems)
- Catalog query
  - ◆ Ad hoc science pipeline processing

Each system will choose an appropriate framework. Extensions to the framework will have to be negotiated.

Each system will need to devote sufficient time to documenting system-unique faults and tolerance mechanisms. Preventive maintenance procedures will also need to be documented.

The end result will be a set of per-system applications of concepts and practices.