Gemini Observatory

GIAPI Software Requirements for Instrument Builders

Version 04

Arturo Núñez

GIAPIBuilderReq-01302009/04
Created: January 30, 2009
Modified: November 18, 2014



Change Record

Version	Date	Author	Notes
1	January 30, 2009	Arturo Núñez	• First Version
2	April 26, 2011	Arturo Núñez	• Updates for new instruments
3	November 16, 2014	Arturo Núñez	• Updated OS preferred choice. Added performance requirement for sequence commands.
4	November 18, 2014	Arturo Núñez	• Updated REQ-GIAPI-SC-19 after review by Systems Engineering.

Table of Contents

1 Introduction *1*

1.1 Document Purpose 1

1.2 Intended Readership 1

1.3 Acronyms 1

1.4 Reference Materials 2

2 Requirements *2*

2.1 Gemini Instrument layout 3

2.2 Hardware Requirements 4

2.3 Operating System Requirements 6

2.4 GIAPI Library Use 6

2.5 Status, Alarms and Health 7

2.6 Sequence Commands Requirements 9

2.7 Gemini Systems Interaction Requirements 12

2.8 Requirements to use Gemini Services 13

2.9 Data Production Requirements 13

1 Introduction

This document provides a list of the instrument builder software requirements for teams using the Gemini Instrument Application Programmer Interface (GIAPI). The GIAPI concept was introduced in 2004 to support the second generation of Gemini instruments, the products of the Aspen instrument process[1].

The Gemini Software instrument documentation tree is shown in Figure 1. This document is shown surrounded by a box. The document you are reading is a companion document to the top-level document *GIAPI Design and Use*[2]. GIAPI Design and Use is the document that describes how the GIAPI works and how builders should use the library to integrate with Gemini software and hardware systems. Requirements and patterns for its use are described to clarify how responsibilities are balanced between Gemini and the instrument builder.

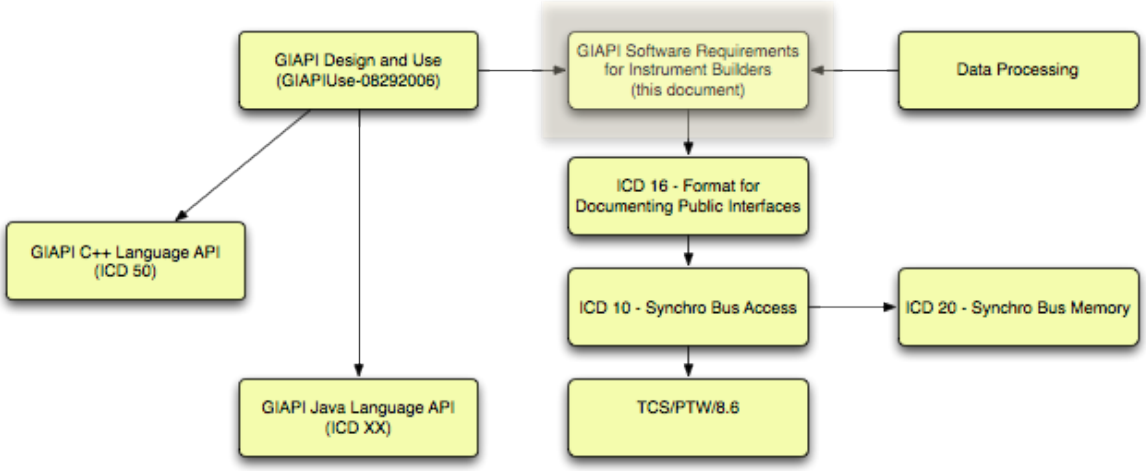


Figure 1: GIAPI Documentation Tree

The GIAPI provides a client integration library in two computer programming languages: C++ and Java. This document presents the requirements for the instrument builders to construct software that will interface with Gemini via the GIAPI.

1.1 Document Purpose

The purpose of this document is to list the software requirements to build instrument software that is compatible with the GIAPI. The requirements are extracted from the *GIAPI Design and Use* [2] Each requirement listed in this document has a unique ID that will be used to uniquely identify the requirement in all the stages of the instrument development process.

1.2 Intended Readership

The intended audience for this document is *groups who are writing software or design review documents for Gemini instruments*. The document is targeted specifically towards software managers, software engineers or other specialists who are responsible of design, implement, integrate or test a Gemini instrument or need to estimate the size and cost of the software component of a Gemini instrument.

1.3 Acronyms

ATEUI	Acceptance Test and Engineering User Interface
CC	Components Controller
DC	Detector Controller
ECC	Error Correcting Code
EPICS	Experimental Physics and Industrial Control System
GIAPI	Gemini Instrument Application Programmer Interface
GSA	Gemini Science Archive

GDSN	Gemini Data Storage Network
ICD	Interface Control Document
IS	Instrument Sequencer
NEBS	Network Equipment Building
NIC	Network Interface Card
OLDP	On Line Data Processing
SDD	Solid State Drive
TCS	Telescope Control System
TLC	Top Level Computer
WCS	World Coordinate System

1.4 Reference Materials

[1] Guidelines for Designing Gemini Aspen Instrument Software, Kim Gillies, AspenSoft-03072004-6

[2] *GIAPI Design and Use*, Kim Gillies, Arturo Núñez, GIAPIUse-08292006.

[3] FITSIO Home Page, http://heasarc.gsfc.nasa.gov/docs/software/fitsio/fitsio.html

[4] FITS Standard Specifications, http://fits.gsfc.nasa.gov/fits_standard.html

2 Requirements

One of the important goals of this document is to ensure that builders know what behavior is required and expected of their instrument software to enable integration with Gemini's software system. Throughout the *GIAPI Design and Use* [2] document the text has referenced builder requirement, and they are all listed here in one place for easier reference.

The following sections present the requirements for the instrument software that are to form the basis to build a Gemini compliant instrument that interface with Gemini via the GIAPI. The requirements should be used to evaluate the appropriateness of related systems, and ultimately they form the basis of the instrument software design.

Each guideline or requirements is presented in the following type of table:

REQ-GIAPI-SYS-XX

Name:	The name of the requirement or guideline is described here.
Description:	Notes needed for clarification are here (optional).
Source:	Where does this requirement come from?

The title of the item means **Req**uirement **GIAPI-Sys**tem-XX. Here **SYS** is replaced with a short code (two or three letters) to represent the aspect of the system the requirement is involved with, such as hardware, library use, status generation, data production, etc. The description contains any necessary clarifying information. The source entry traces this requirement to the original source if available.

Table 1 summarizes the different categories and the corresponding codes we will use through the document to classify the different builder requirements. In addition, the table shows the section where the requirements corresponding to each category are presented.

Short Code	Description	Section
INS	Requirements associated to one of the logical components of a Gemini instrument: Instrument Sequencer, Detector Controller or the Components Controller.	2.1
HW	Hardware requirements.	2.2

Short Code	Description	Section
OS	Operating System requirements.	2.3
USE	GIAPI general use requirements.	0
STA	Requirements to produce status, alarms and health information to Gemini via the GIAPI.	2.5
SC	Requirements to receive and process the Gemini Sequence Commands.	2.6
GI	Requirements to interact with Gemini Systems	2.7
SVC	Requirements to use the services provided by the GIAPI	2.8
DP	Data production requirements	2.9

Table 1: Categories used to classify Builder Requirements

The following sections present the requirements for each one of these categories.

2.1 Gemini Instrument layout

A typical Gemini instrument consists logically of three parts: the Components Controller (CC), Detector Controller (DC), and Instrument Sequencer (IS). This section describes the requirements associated to these parts.

REQ-GIAPI-INS-01

Name:	Handle sequencing details internally
Description:	The IS must handle all internal sequencing details of the instrument and not expect that responsibility to be handled by the sequencing software in the OCS
Source:	GIAPI Design and Use, section 3.1

REQ-GIAPI-INS-02

Name:	Move devices and motors from a safe state to another
Description:	The Components Controller must move the configuration of the instrument devices and motors from one safe state to another
Source:	GIAPI Design and Use, section 3.2

REQ-GIAPI-INS-03

Name:	Optimize multiple motions
Description:	Complex configurations resulting in multiple motions should be applied in parallel. If several different components undergo movement with one applied configuration, the devices should be moved in parallel to minimize the time needed to reconfigure the instrument.
Source:	GIAPI Design and Use, section 3.2

REQ-GIAPI-INS-04

Name:	Handle motion peculiarities within the instrument
Description:	The components controller must handle any motion peculiarities within the

	instrument. Instruments with multiple filter wheels can have dependencies between wheels. For example, if wheel 1 is moved from A to G, wheel 2 must be also be moved to a blocking position such that no bright light ever hits the detector. Then following the conclusion of the filter 1 move, filter 2 must be moved back to its previous position. This kind of instrument-specific logic, if needed, should be embedded in the components controller logic
Source:	GIAPI Design and Use, section 3.2

REQ-GIAPI-INS-05

Name:	Avoid unnecessary movements
Description:	The components controller must be sufficiently knowledgeable about the current motor and device configuration such that unnecessary motions or movements are eliminated
Source:	GIAPI Design and Use, section 3.2

REQ-GIAPI-INS-06

Name:	Preprocessing of data needs to be provided by the detector controller
Description:	The Detector Controller must provide any preprocessing of data that may be needed before forwarding the data products to Gemini.
Source:	GIAPI Design and Use, section 3.3.1

REQ-GIAPI-INS-07

Name:	Do not rely on the Gemini pipeline to do data processing.
Description:	Well-formed, easily viewable datasets must be passed to Gemini. The instrument should not rely on the Gemini pipeline to do this kind of processing.
Source:	GIAPI Design and Use, section 3.3.1

2.2 Hardware Requirements

These requirements describe the hardware choices available for Aspen Instruments.

REQ-GIAPI-HW-01

Name:	64-bit CPU(s)
Description:	System should use AMD or Intel 64-bit capable x86 CPU(s) using multiple processors and/or multi-cores as needed[1].
Source:	

REQ-GIAPI-HW-02

[1] This specification may change over time to reflect the state of available hardware at a future date.

Name:	Redundant power supply
Description:	The computers should have redundant power supplies. The important requirement is that the computer not goes down when one of the power supplies fails and that no human intervention is needed to switch to the backup.
Source:	

REQ-GIAPI-HW-03

Name:	Redundant, accessible disk drives
Description:	Disk drives should be redundant allowing a RAID setup. Disks should be easily accessible from the front and allow hot-pluggable exchange without opening the chassis. Solid-state drives (SDD) such as the Flash memory-based drives should be considered for drives within the dome environment.
Source:	

REQ-GIAPI-HW-04

Name:	Error correcting memory
Description:	Systems should use registered ECC memory.
Source:	

REQ-GIAPI-HW-05

Name:	Add-in board support
Description:	Systems should support at least 2 full-height PCI-X or PCI Express slots[1].
Source:	

REQ-GIAPI-HW-06

Name:	Dual Ethernet ports
Description:	Systems should provide at least 2 Ethernet 100/1000 Ethernet NICs[1].
Source:	

REQ-GIAPI-HW-07

Name:	USB-2.0 ports
Description:	The system should provide at least one USB 2.0 port to allow booting from a USB-based memory device[1].
Source:	

REQ-GIAPI-HW-08

Name:	NEBS compatibility
Description:	Given the harsh enclosure environment and extreme conditions within the telescope

	enclosure NEBS Level 3 compatible or carrier-grade equipment is preferred unless systems are mounted in the computer room.
Source:	

REQ-GIAPI-HW-09

Name:	Document hardware choices
Description:	Hardware choices and physical layout for computer control of the instrument must be designed, documented and justified as part of the instrument design process. Solutions that stray far from recommendations must be justified during reviews
Source:	GIAPI Design and Use, section 5.1

REQ-GIAPI-HW-10

Name:	Favor homogeneity in computers
Description:	All instrument computers should follow the TLC specifications unless there is a significant reason not to do so.
Source:	GIAPI Design and Use, section 5.2

REQ-GIAPI-HW-11

Name:	GIAPI to run on hardware that meets the same specifications as the TLC
Description:	Distributing GIAPI functionality to other computers besides the TLC requires hardware that meets the same specifications as the TLC
Source:	GIAPI Design and Use, section 5.2

2.3 Operating System Requirements

Requirements related to the operating system for an Aspen Instrument are listed here.

REQ-GIAPI-OS-01

Name:	TLC computer runs the Linux Operating System
Description:	The TLC computer runs the Linux operating system. At this time, the specified version for this type of application at Gemini is CentOS 6, 64-bits This specification is likely to change given the rapid changes in Linux and hardware, but changes in Linux OS should not impact builders.
Source:	GIAPI Design and Use, section 6

10

2.4 GIAPI Library Use

These requirements describe GIAPI-wide requirements

REQ-GIAPI-USE-01

Name:	Status and command naming convention
Description:	Every status or command item that passes between the instrument and Gemini must have a unique name within the entire Gemini control system using the following convention: The item name consists of the instrument ID followed by a component name and a field name. The field portion of the item name is separated from the component name by a period. A colon separates all other parts of the item name. The total length of the item name must be 60 characters or less. This length limit is determined by a limit in EPICS (version 3.14 and later).
Source:	GIAPI Design and Use, section 8.1

REQ-GIAPI-USE-02

Name:	Status and command parameter and attributes defined in the public interface
Description:	The status and command parameter and attribute names are part of the public interface of the instrument and are agreed upon during the instrument design phase to ensure that the correct instrument information is available.
Source:	GIAPI Design and Use, section 8.1

2.5 Status, Alarms and Health

Requirements about producing status information to Gemini are listed here.

REQ-GIAPI-STA-01

Name:	Provide state and health to Gemini through the GIAPI
Description:	An instrument must provide information describing its state and health to Gemini through the GIAPI
Source:	GIAPI Design and Use, section 9.1

REQ-GIAPI-STA-02

Name:	Track and raise alarms
Description:	Instruments must keep track of alarm conditions in the instrument and make that information available by raising an alarm when needed
Source:	GIAPI Design and Use, section 9.3

REQ-GIAPI-STA-03

Name:	Provide a status called INST:health
Description:	The instrument must provide a pre-defined kind of status called *INST:health* to report overall instrument health. INST is the instrument prefix agreed upon unique name of the instrument. Other health values for subsystems can be published as well, but are not required
Source:	GIAPI Design and Use, section 9.4

REQ-GIAPI-STA-04

Name:	Health must be kept up to date due to any changes in status items
Description:	It is up to the instrument developers to determine when their instrument is in each health state. The typical implementation is one where individual subsystems or devices each define a health value and the instrument health value is recursively defined in terms of the children values
Source:	GIAPI Design and Use, section 9.4

REQ-GIAPI-STA-05

Name:	Consider and document the interrelationships of important status items
Description:	The builder must consider and document the interrelationships of important status items. There can obviously be a relationship between health, status, and alarms that instrument builders must consider and document
Source:	GIAPI Design and Use, section 9.4

REQ-GIAPI-STA-06

Name:	Builder code should only post updates to status items that change
Description:	
Source:	GIAPI Design and Use, section 9.5.1

REQ-GIAPI-STA-07

Name:	Status publishing latency
Description:	The instrument must publish status values to the GIAPI within 300 ms of being changed
Source:	GIAPI Design and Use, section 9.5.5.1

REQ-GIAPI-STA-08

Name:	Status updates while changing
Description:	A status item that is changing continuously over several seconds should be updated at least every 1000 ms to and no faster than once every 500 ms to ensure the user is kept up to date
Source:	GIAPI Design and Use, section 9.5.5.2

REQ-GIAPI-STA-09

Name:	No status value changes can be lost or missed.
Description:	For instance, in the situation where a filter goes from A to D, only publishing the final position D is not acceptable
Source:	GIAPI Design and Use, section 9.5.5.3

REQ-GIAPI-STA-10

Name:	Status values must always be published in the order in which they occur.
Description:	For instance, in the situation where a filter goes from A to D, status must be published as A, B, C, D, not A, C, B, D or some other order
Source:	GIAPI Design and Use, section 9.5.5.4

2.6 Sequence Commands Requirements

This section presents the requirements associated to handling Gemini sequence command by an Aspen compliant instrument.

REQ-GIAPI-SC-01

Name:	Self-tests documented in the public interface
Description:	The **Test** sequence command implies that an instrument must have a set of self-tests that are documented in the public interface
Source:	GIAPI Design and Use, section 10.3.1

REQ-GIAPI-SC-02

Name:	Document what devices and systems are affected by each of the setup and maintenance sequence commands
Description:	Builders need to document in the instrument's public interface what devices and systems are affected by each of the setup and maintenance sequence commands
Source:	GIAPI Design and Use, section 10.3.1

REQ-GIAPI-SC-03

Name:	Provide the instrument startup script written as a traditional Linux init script. This script handles the reboot functionality as well.
Description:	The instrument builder must provide the instrument startup script written as a traditional Linux init script (preferably in bash or sh) that will take the traditional init.d arguments of: start, stop, restart, status, usage, and reboot. The name of the script should be the agreed upon name of the instrument followed by .sh (e.g. gpi.sh). This script acts upon the entire instrument software system, including instrument software on the TLC and any subsystems. The *instrument startup script* is used to handle the **reboot** functionality
Source:	GIAPI Design and Use, section 10.3.2

10

REQ-GIAPI-SC-04

Name:	The instrument must be capable of executing changes to its configuration during verify mode

Description:	
Source:	GIAPI Design and Use, section 10.3.3

REQ-GIAPI-SC-05

Name:	Behavior for instruments that cannot pause
Description:	If the instrument can not pause, **pause** should be implemented to throw an error and **continue** should successfully complete immediately
Source:	GIAPI Design and Use, section 10.3.5

REQ-GIAPI-SC-06

Name:	An instrument should support **pause** if possible
Description:	
Source:	GIAPI Design and Use, section 10.3.5

REQ-GIAPI-SC-07

Name:	Document pause and continue behavior
Description:	The instrument's behavior for pause and continue should be documented in its public interface
Source:	GIAPI Design and Use, section 10.3.5

REQ-GIAPI-SC-08

Name:	Document each sequence command as part of the instrument's public interface
Description:	The description of the implementation of each sequence command is part of the description of the instrument's public interface
Source:	GIAPI Design and Use, section 10.3.6

REQ-GIAPI-SC-09

Name:	All instruments must be capable of receiving and responding to all the sequence commands.
Description:	Some of the commands may not be relevant for the instrument. In that case, it must acknowledge the command and indicate immediate successful completion
Source:	GIAPI Design and Use, section 10.3.6

REQ-GIAPI-SC-10

Name:	Sequence commands must be accepted at all times
Description:	At no time should the instrument enter a state where it cannot or does not accept a command. Accepting means processing a command request
Source:	GIAPI Design and Use, section 10.4

REQ-GIAPI-SC-11

Name:	**init** to be executed at any time
Description:	The instrument should allow **init** to be executed at any time resulting in the end of any ongoing sequence commands and the beginning of the **init**. This is needed to clear any problems that might occur in execution of actions related to other sequence commands
Source:	GIAPI Design and Use, section 10.4

REQ-GIAPI-SC-12

Name:	Refuse to start the associated actions of another concurrent sequence command received, except for **init** and **apply**
Description:	If executing actions from a sequence command other than **init** or **apply**, the instrument should refuse to start the associated actions of another concurrent sequence command received.
Source:	GIAPI Design and Use, section 10.4

REQ-GIAPI-SC-13

Name:	Support concurrent execution of **apply** commands.
Description:	An instrument must support the concurrent execution of multiple **apply** commands.
Source:	GIAPI Design and Use, section 10.4

REQ-GIAPI-SC-14

Name:	Optimize handling of concurrent **apply** commands
Description:	When receiving concurrent **apply** sequence commands, the instrument must handle the case where **apply** is to a device or subsystem that is already being demanded to a new configuration
Source:	GIAPI Design and Use, section 10.4

REQ-GIAPI-SC-15

Name:	Only one **observe** will execute at a given time.
Description:	The instrument should assume that only one **observe** will execute at a given time, and an error should be returned if an **observe** is received while executing **observe**
Source:	GIAPI Design and Use, section 10.4

REQ-GIAPI-SC-16

Name:	Execute ***preset*** for any sequence command at any time
Description:	A system must be able to execute a ***preset*** for any sequence command at any time. The code that implements the handle method must evaluate requests and dispatch actions. It cannot execute actions directly/synchronously and become unavailable

	while busy
Source:	GIAPI Design and Use, section 10.6

REQ-GIAPI-SC-17

Name:	***preset*** activity must still execute when actions of that command are already executing
Description:	It is possible that a sequence command will be received when actions related to that command are already executing. The acceptance phase or ***preset*** activity must still execute
Source:	GIAPI Design and Use, section 10.6

REQ-GIAPI-SC-18

Name:	Builders must register an implementation of `SequenceCommandHandler` for each of the sequence commands
Description:	
Source:	GIAPI Design and Use, section 10.6

REQ-GIAPI-SC-19

Name:	A `HandlerResponse` is returned whenever the `handle` method is invoked, and it must be returned within 300ms (REQ-GIAPI-SC-19). Failure to do so will result in the OCS considering the system unresponsive.
Description:	Use of the correct `HandlerResponse` is very important in order to meet the 300ms threshold. For instance, different `HandlerResponse` must be used for Short versus Long commands. Refer to Table 16 in the GIAPI Design & Use document for full details.
Source:	GIAPI Design and Use, section 10.6

2.7 Gemini Systems Interaction Requirements

The following are the requirements to interact with specific Gemini systems.

REQ-GIAPI-GI-01

Name:	Monitoring allowed EPICS status items
Description:	The names of the EPICS status items that are possible to monitor via the GIAPI are agreed upon during the instrument design as part of the Instrument/Gemini interface
Source:	GIAPI Design and Use, section 11.1.1

10

REQ-GIAPI-GI-02

Name:	Produce WCS information

Description:	To get WCS information for dataset headers, the TCS context information provided by the GIAPI must be used. There is no plan to recode the C libraries into any other language so complex transformation will need to be done in C
Source:	GIAPI Design and Use, section 11.4

REQ-GIAPI-GI-03

Name:	Use the Gemini-provided Syncro-bus library to control the Gemini Secondary.
Description:	All instruments are required to use the Gemini-provided Syncro-bus library to control the Gemini Secondary. This library is a Linux-based C-level library/driver to access the Synchro Bus.
Source:	GIAPI Design and Use, section 11.5

2.8 Requirements to use Gemini Services

The GIAPI provides a set of services to the instrument software. These are the requirement associated to them.

REQ-GIAPI-SVC-01

Name:	Use the GIAPI services
Description:	When an instrument needs a service like one provided by the GIAPI, the GIAPI service should be used. The GIAPI provides logging, observatory time access, and GIAPI properties as services to the instrument.
Source:	GIAPI Design and Use, section 12

2.9 Data Production Requirements

This section presents the requirements associated to data production for an Aspen instrument

REQ-GIAPI-DP-01

Name:	Instruments must write datasets in two modes: engineering/testing and operations.
Description:	In both modes, instruments write datasets in FITS format using what should be identical code. Having a separate engineering mode allows builders to create headers for testing that are appropriate for their needs
Source:	GIAPI Design and Use, section 13.2

10

REQ-GIAPI-DP-02

Name:	The FITS files written by an instrument must conform to the FITS standard [4]
Description:	
Source:	GIAPI Design and Use, section 13.2.1

REQ-GIAPI-DP-03

Name:	Use the CFITSIO library.

Description:	Instrument builders are required to use the CFITSIO library [3]. Files will need to be specified and written to use the capabilities of this standard library.
Source:	GIAPI Design and Use, section 13.2.1

REQ-GIAPI-DP-04

Name:	Write datasets with a minimal header unit and any minimal extension header units needed by the FITS standard.
Description:	Datasets should be written at Gemini with a minimal header unit and any minimal extension header units that are needed as required by the FITS standard. The contents of this header are determined during the design process for the instrument
Source:	GIAPI Design and Use, section 13.2.2

REQ-GIAPI-DP-05

Name:	Control engineering/operations mode FITS headers through the ATEUI and instrument configuration
Description:	If the instrument software chooses to support an engineering mode FITS header in addition to the operations mode FITS header, it must be possible to control this through the ATEUI and instrument configuration
Source:	GIAPI Design and Use, section 13.2.3

REQ-GIAPI-DP-06

Name:	Write datasets directly to the Gemini Data Storage Network.
Description:	Instruments will write their datasets directly to the GDSN. A storage volume is mounted on one of the instrument computers; either the TLC or another computer in the instrument computing environment
Source:	GIAPI Design and Use, section 13.2.4

REQ-GIAPI-DP-07

Name:	Storage for intermediate results, configuration, or other non-data files or temporary files should be kept on storage local to the instrument; the GSDN is used only for science datasets or ancillary files
Description:	
Source:	GIAPI Design and Use, section 13.2.4

REQ-GIAPI-DP-08

Name:	Do not depend on processing systems outside of the instrument.
Description:	The builder should not depend on the Gemini pipeline or processing systems outside of the instrument to apply algorithms to unscramble data from detectors or to assemble data into complete and correct datasets from the data of individual detectors. The details of this kind of processing are to remain within the instrument software
Source:	GIAPI Design and Use, section 13.2.4

REQ-GIAPI-DP-09

Name:	Header content and data unit format of ancillary files
Description:	Any ancillary files that need to be associated with science data and possibly stored in the GSA must follow identical procedures as with science data for defining header content and data unit format
Source:	GIAPI Design and Use, section 13.2.4

REQ-GIAPI-DP-10

Name:	Status items to publish for science headers
Description:	Gemini and builders agree upon the instrument status items that must be publicly available for science headers
Source:	GIAPI Design and Use, section 13.3

REQ-GIAPI-DP-11

Name:	Gemini and builders must agree upon the structure of the data part of the dataset FITS file
Description:	
Source:	GIAPI Design and Use, section 13.3

REQ-GIAPI-DP-12

Name:	Order of observation events
Description:	Observation Events must happen in the order specified in table 17 and Figure 21 of the GIAPI Design and Use document [2]
Source:	GIAPI Design and Use, section 13.4.2

REQ-GIAPI-DP-13

Name:	Timing of observation events
Description:	The instrument code must be constructed to allow the observation events to be fired at the correct times in an execution of the **observe** sequence command, as specified in Table 17 of the GIAPI Design and Use document [2].
Source:	GIAPI Design and Use, section 13.4.2

REQ-GIAPI-DP-14

Name:	Use FITS format for ancillary files
Description:	All ancillary files will be written as FITS format files using the same software library as science data
Source:	GIAPI Design and Use, section 13.5

REQ-GIAPI-DP-15

Name:	Content and use of ancillary files
Description:	The content and uses for the ancillary files must be agreed upon during the instrument design phases as part of the negotiation on header content and data unit format
Source:	GIAPI Design and Use, section 13.5