




A Real-time Control Computer for the E-ELT

**Document: GF-PDR-08
Interface Definition Document**

**Version 1.1
17th January 2016**

Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 2 of 12
Interface Definition Document		

Change Record

Version	Date	Author(s)	Remarks
0.1	5 Jan 2016	Eddy Younger	Initial version
0.2	8 Jan 2016	Eddy Younger	Comments from Damien Gratadour
0.3	11 Jan 2016	Eddy Younger	Comments from Hugues Deneux
0.4	16 Jan 2016	Eddy Younger	Updated system architecture diagram (fig. 1)
1.0	17 Jan 2016	Eddy Younger	First release (PDR)



Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 3 of 12
Interface Definition Document		

Table of Contents

1 Scope.....	6
2 System Architecture.....	6
3 Interface design concept.....	7
4 Module Internal Interfaces.....	9
4.1 Publish/subscribe interface.....	9
4.2 Pipeline stream interface.....	9
4.3 Control interface.....	9
5 Real-time Data Pipeline.....	9
5.1 Internal Interfaces.....	10
5.2 External Interfaces.....	10
6 Soft Real-time System.....	11
7 Simulator.....	12
7.1 Internal Interfaces.....	12
7.2 External Interfaces.....	12


Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 4 of 12
Interface Definition Document		

Applicable Documents

No.	Title	Reference	Issue	Date
AD1	Green Flash: System Architecture	GF-PDR-04	1.5	17 Dec 2015

Reference Documents


No.	Title	Reference	Issue	Date
RD1	SPARTA: Software Preliminary Design	VLT-TRE-ESO-16100-4261	1	15 Jun 2007

Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 5 of 12
Interface Definition Document		

Acronyms and abbreviations

Table 1 Acronyms and Abbreviations

AO	Adaptive Optics
CORBA	Common Object Request Broker Architecture
COTS	Comercial off-the-shelf
DDS	Data Distribution Service
DM	Deformable Mirror
DMC	Deformable Mirror Controller
E-ELT	European Extremely Large telescope
ESO	European Southern Observatory
FDR	Final Design Review
(s)FPDP	(serial) Front Panel Data Port
FSM	Fast Steering Mirror
LGS	Laser Guide Star
NGS	Natural Guide Star
PDR	Preliminary Design review
RTC	Real-Time Computer
RTCS	Real-Time Control System
TCS	Telescope Control System
WFS	Wavefront Sensor

Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 6 of 12
Interface Definition Document		

1 Scope

This document defines the software interfaces between the components of Green Flash described in the System Architecture document **AD1** and illustrated in **Figure 1**. This early version of the document serves only to capture and describe the set of interfaces required; the design phase of Green Flash will fully specify the interfaces in terms of their syntax and semantics, and this complete specification will be documented in a version of this document to be completed for FDR. In particular, an important part of Green Flash is to evaluate and subsequently down-select from a number of candidate technologies for implementation of the final prototype; it is not possible to fully specify interfaces until this selection has taken place.

2 System Architecture

The Green Flash system architecture is described in **AD1**, and is similar to the distributed design used for SPARTA (see **RD1**). It consists of 3 systems:

- Hard real-time data pipeline
- Soft real-time system
- Simulator system

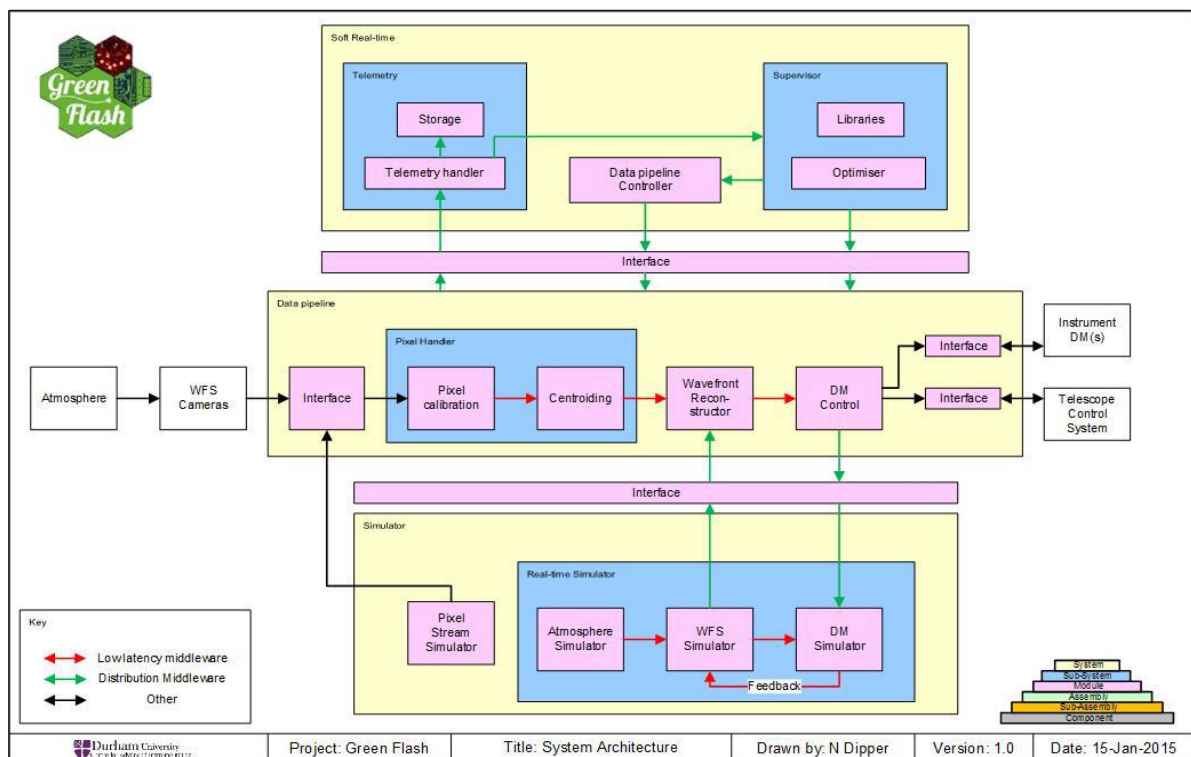



Figure 1: Top-level System Diagram

The top level system engineering diagram for the full system is shown in a very generalised form in Figure 1.

Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 7 of 12
Interface Definition Document		

3 Interface design concept

Figure 2 illustrates the design concept for Green Flash software modules and their interfaces. As far as is practical, the interfaces to a module are abstracted from its core logic into separate components loosely coupled to the core logic via a small number of interfaces.

Conceptually there are three middleware systems in Green Flash:

1. Telemetry middleware

This will be a publish/subscribe system used to distribute telemetry data within the system, with minimum overhead and redundancy/duplication. It must be capable of handling large amounts of data at real-time or close to real-time rates. It is planned that RTI DDS will be used.

2. Command middleware

This is the system by which command, control and configuration information will be communicated. Its requirements are much less demanding than the telemetry middleware. It is planned that CORBA will be used, since it is standardised and well supported by tools and platform/language bindings. The CORBA RPC paradigm is better suited to synchronous command mediation than the publish-subscribe paradigm of DDS since it allows a remote method to return a response value (a value or return code) to the caller; a publish/subscribe system would require the caller to monitor a topic published by the target entity in order to receive a response. The CORBA nameserver will be hosted in the soft real-time system.


3. Low-latency middleware

The data pipeline requires a very low latency, very low jitter data transport between its modules. Minimum complexity and overhead is essential here. Since this is essentially point-to-point transport between functional modules of the pipeline, complex middleware is not required and so a simple high bandwidth low-latency transport will be used, such as shared memory, OpenMPI or simple UDP sockets. The actual transport is highly dependent upon the final design of the prototype: an embedded microserver implementation is likely to use a very different mechanism to a CPU system with accelerators.

Each module contains at least the following components:

1. Core Logic: the core functionality of the module. This may itself be decomposed into a number of functional components.
2. Control Server: the interface by which the module receives command/control instructions from external software entities through the control middleware.
3. Status Publisher: each module publishes its status to the telemetry middleware.

In addition, the module may contain other components:

Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 8 of 12
Interface Definition Document		

1. Control client(s): if the module needs to command or control other modules in the system, it will contain a client for each such module. The Control Clients interface with the Control Server in the module which is controlled, via the control middleware.
2. Telemetry publisher(s): the module may publish one or more telemetry topics in addition to the mandatory status/state topic. A publisher is required for each such topic.
3. Telemetry subscriber(s): the module may make use of telemetry information published by other modules . A subscriber component is required for each topic to which the module subscribes.
4. Stream interfaces: modules in the data pipeline will consume and produce streaming data which is communicated between modules via the low-latency middleware (see Figure 1). The interface between this middleware and the core logic is implemented by the stream interface components.

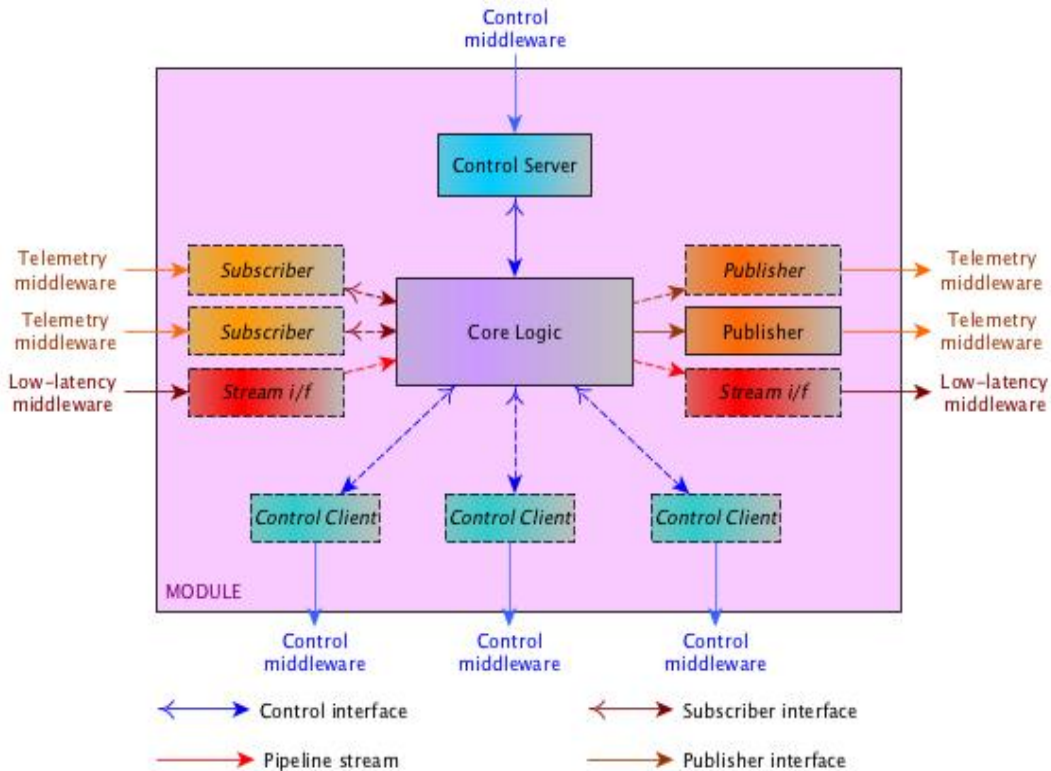



Figure 2: High-level module interface design

Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 9 of 12
Interface Definition Document		

Dotted lines/outlines and italics indicate components which are optional depending on the nature of the module. Every module will contain at least a control server, and a publisher for status telemetry.

4 Module Internal Interfaces

Figure 2 illustrates the three major inter-component interfaces of a module: Publish/Subscribe (telemetry), Control, and Pipeline stream (data pipeline modules only).

4.1 Publish/subscribe interface

The telemetry middleware will transport items of telemetry data identified by **Topic**, which will uniquely identify the type and source of the information. This concept (and name) is integral to DDS, but this does not preclude the use of another middleware as the concept is necessary in any Publish/subscribe system. Publishers each publish, and subscribers subscribe to, a single Topic. Topics will include status information (Reconstructor Status, WFS Camera1 Status,...) and real-time pipeline telemetry (Raw- and Calibrated Pixels, Centroids, M4 commands,...). The set of Topics, and their associated data types, will be static and defined system-wide. Subscribers will be able to specify a decimation rate, and to modify this on-the-fly.

4.2 Pipeline stream interface

Data pipeline streams conceptually consist of frames of data of fixed type (typically simple scalar types such as signed or unsigned short ints, bytes, etc). Each frame begins with a header containing a start-of-frame indicator, the frame length, and a frame-id analogous to the telemetry Topic, which uniquely identifies the stream.

Streams differ from telemetry in that partial frames may be transmitted and received; this potentially improves latency since a pipeline module may begin to process a frame before the complete frame has been received. For example, a MVM-based reconstructor may in principle begin calculation as soon as the first centroid of a new frame is received. The atomicity of a stream source may be as small as a single frame element. The stream consumer interface will allow a consumer to request a specific number of frame elements from the middleware/source, but must also be able to deal with the cases where a) fewer than requested elements are received, where a receive timeout was specified.


4.3 Control interface

The control interface is highly specific to the core logic of the module. In each case it must support synchronous and asynchronous (long-running) commands without blocking.

5 Real-time Data Pipeline

This is the low latency, low jitter data pipeline. The system consists of four separate modules:

- Pixel Calibration

Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 10 of 12
Interface Definition Document		

- Centroiding
- Wave-front reconstruction
- DM controller

5.1 Internal Interfaces

1. Inter-module interface

These are pipeline stream interfaces mediated by the low-latency middleware, transporting pixels, centroids, reconstructed wavefronts and DM commands between the pipeline modules.

5.2 External Interfaces

1. Wave-front sensor camera interfaces

This interface carries pixel data from the camera to the pixel handling module. The interface to a particular type of camera is hardware-specific; Green Flash will support cameras using GigE-Vision (GbE and 10GbE). The WFS Camera interface shall accept camera input and produce output as a pipeline stream to the low-latency middleware which couples it to the Pixel Calibration module. Whilst the Pixel Calibration module is a part of the real-time data pipeline, all or part of the pixel handling may be encapsulated in an 'intelligent interface' or “smart camera”.


Support for other camera interfaces such as CameraLink, AIA and sFPDP is beyond the scope of Green Flash, however the camera interface will be designed such that future support for other camera types is not precluded, perhaps via interface conversion hardware.

2. Interface to the telescope M4 DM

The telescope M4 interface definition is not yet available. The output of the real-time data pipeline will be in the form of DM demand data that conforms to the definition of this interface. This will be a bi-directional interface as the telescope M4 systems will produce feedback to the real-time data pipeline.

3. Interface to instrument DMs

This module shall receive streaming input from the low-latency middleware and transform this into a form appropriate to the instrument-specific DM(s). DMs incorporated within instruments are expected to be COTS assemblies; manufactures generally provide device-drivers and API libraries for a small number of common operating systems/computer platforms. To drive the DM directly from accelerator

Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 11 of 12
Interface Definition Document		

hardware will require that the (possibly proprietary) interface is implemented in the accelerator.

4. Interface to the soft real-time system

- i. The Soft real-time systems will subscribe to telemetry data published by the data pipeline via the telemetry middleware.
- ii. Commands and calibration/configuration/optimisation data will be sent via the command middleware to the data pipeline.

5. Interface to simulator

The simulator will provide a facility to inject simulated data into the real-time data pipeline at various stages, and to receive data from the actual real-time data pipeline into a simulated DM module. These interfaces will be pipeline streams to correspond with the inter-module interfaces in the data pipeline, allowing data to be injected into or received from the pipeline without changes to the pipeline itself.

6 Soft Real-time System


The soft real-time system is responsible for the configuration, calibration, control and optimisation of the hard real-time pipeline. It consumes telemetry data from the data pipeline (and potentially other parts of a complete AO system, such as mechanisms), and uses this data to calculate optimised parameters for the pipeline.

1. Interface to the real-time data pipeline

This is a two-directional interface. Telemetry data is published from the real-time data pipeline to the soft real-time system. Command data is sent to the real-time data pipeline from the soft real-time system for configuration, calibration, control and optimisation to the pipeline, for example the download of reconstructor- or pixel calibration parameters

2. User Interface

All control interfaces to the full system are connected via the soft real-time system and implemented via a 'supervisor' module. This module will contain a number of Control client components to interface with the rest of the system via the control middleware.

Observatoire de Paris Durham University Microgate PLDA		Title: Interface Definition Document Ref: GF-PDR-08 Version: 1.1 Date: 17 January 2016 Authors: Eddy Younger Nigel Dipper Page: 12 of 12
Interface Definition Document		

7 Simulator

The proposed simulator system (see **Figure 1**) is intended to simulate elements of the AO system that are not physically present, producing and/or consuming the appropriate data and simulating the effect of the hardware in response. This includes simulation of:

- The atmosphere (phase screens)
- Camera pixel data
- Wave-front sensors
- Deformable mirrors.

Pixel data simulation allows cameras to be tested that are not present or may not yet exist! The data stream must be deterministic to correctly represent a camera pixel stream.

The other modules simulate parts of the AO system hardware or data pipeline. Existing AO simulation code generally runs at less than real-time rates. Such code can be accelerated to close to the target rates of the data pipe-line using accelerator technology.

It shall be possible to run the data pipeline with none, one or several aspects in simulation.

7.1 Internal Interfaces

Internal interfaces between modules of the simulator are pipeline streams.

7.2 External Interfaces

1. Camera simulator interface to the data-pipeline pixel handler input;
2. WFS simulator interface to the data pipeline reconstructor input
3. DM simulator interface to the DM controller output

Each of these is an example of a pipeline stream interface; the first two being sources of respectively simulated pixels and simulated centroids, and the latter a consumer of DM commands.

4. Telemetry

The simulator modules must produce telemetry in the same form (i.e. publish the same telemetry topics) as the data pipeline modules which they simulate (where applicable).

5. Control

The simulator must include Control server(s) to enable the User Interface and Soft real-time systems to configure and control it.