Heliophysics Integrated Observatory

Project No.: 238969
Call: FP7-INFRA-2008-2

HELIO Architecture
Version 2.2

Title:	**HELIO Architecture**
Document No.:	HELIO_FHNW_S1_001_RP Deliverable: **S1.1b**
Date:	31 Aug 2011
Editor:	**A. Csillaghy**, FHNW
Contributors:	**G. Pierantoni**, TCD **M. Soldati**, FHNW **S. Felix**, FHNW **K. Benson**, MSSL **A. Le Blanc,** UNIMAN D. Fellows, UNIMAN
Distribution:	Project

CAPACITIES

e-infrastructure

Revision History

Version	Date	Released by	Detail
V1.0	17/12/09	G. Pierantoni	1st Draft
V1.1	12/01/10	G. Pierantoni	2nd Draft • Changes in the introduction, • Changes in the overall design • Included the draft release 1 document • Removed the parts copied from the ***Release Strategy Document***
V1.2	14/02/10	A.Csillaghy	3rd Draft • Corrections and comments
V1.4	01/06/10	G. Pierantoni	4th Draft • Various corrections • Added contributions from services' responsible.
V1.5	05/06/10	A.Csillaghy	5th Draft • References
V 1.6	25/06/10	G. Pierantoni	6th Draft • Minor corrections
V 1.7	29/03/11	G. Pierantoni	7th Draft • Major updates
V 1.8	30/03/11	A. Le Blanc	8th Draft • Additions to Workflow and Taverna workbench section • Changed references to enable cross-references
V 2.0	24/06/11	M. Soldati	Complete revision for new deliverable
V 2.1	24/06/11	R.D.Bentley	Minor updates
V 2.2	08/07/11	M. Soldati	Rework of Section 1-3
V2.2 plusTaverna	29/07/11	D. K. Fellows	Added section on Taverna Server, minor adjustments for other Taverna-related sections
V2.2 [plus hqi,cxs,hrs]	3/08/11	K. Benson	Added to section 6 the HQI, CXS, and HRS sections.
V2.3	31/8/11	A. Csillaghy	2nd iteration on Section 1-3
V2.4	3/09/11	M. Soldati	Rework of section 4 to match section 3

This document will be reissued as work on the HELIO system progresses and our ideas and understanding evolve.

Table of Contents

Table of Figures

Table of Tables

List of Acronyms

API	Application Programming Interface
CIS	Community Interaction Service
HELIO	Heliophysics Integrated Observatory
HFE	HELIO Front End
HPS	HELIO Processing Service
HSS	HELIO Storage Service
SOA	Service Oriented Architecture

1 HELIO as a Service Oriented Architecture

This document defines the architecture of the HELIO infrastructure by describing its components and their interactions. We call the component description the structural architecture of the system, and the interaction description its behavioural architecture.

The architecture of the HELIO infrastructure is based on the concepts of a Service Oriented Architecture (SOA). SOAs feature a set of loosely coupled components. These components offer a public interface called service. A service consists of a collection of service methods, which implement its functionality. The service methods are typically stateless, that means two successive calls to a service method are completely independent.

Using a SOA has two main advantages for HELIO. First, the components can be developed independently at different locations by different teams. Therefore, the services can be hosted by the teams who are experts on the specifically served data. This supports the distributed nature of the project consortium. Second, the components can be deployed redundantly at different locations. This increases the overall stability of the system. In addition, it reduces the need to transport large amount of data as the resource intensive services can be installed close to the data sets.

Intention and structure of this document

This document is intended for readers with a technical background (i.e. software developers) and should serve them as main starting point to understand the system. It is the second part of three documents. It follows the "Concepts Document" [2], from which it takes the conceptual characteristics of the system, and precedes the "Release Strategy Document" [3], to which it delivers the high level structure of the system.

The document focuses on the overall architecture of the system, and leaves the in-depth technical discussion of the components to additional documents, referenced in the text whenever necessary. Furthermore, we aim at shifting technical details closer to the source-code, therefore JavaDoc comments and Unit test classes can also be considered as an extension of this documentation.

This document is structured into several parts:

1. Section 2 gives an overview of the HELIO infrastructure and a brief textual description of the architecture. It is intended for novice and non-technical users.
2. Section 3 describes the structural architecture. This introduces the main components of the infrastructure and outlines how they are related to each other.
3. Section 4 provides the behavioural architecture. This shows how selected components interact with each other.
4. Section 5 details selected services and components provided by HELIO.

2 The HELIO e-Infrastructure in brief

2.1 The solution to a generic problem

The way heliophysicists work with data can be described in a fairly general way, that we call "the basic problem." It can be subdivided into four steps.

In the first step, they need to identify interesting things to study: For instance, they might be interested in solar flares of a specific size, which are associated with a coronal mass ejection. This first search will base solely on metadata -- information that describes observations, or on derived products -- data that has been extracted from observations. In the following, we call these two kinds of data products metadata. Metadata searches will base most of the time on either event lists or on feature lists used as primary selection criteria. Therefore, a first category of services need to take care of this first search functionality.

In the second step, heliophysicists will search for instruments that have observed the event or feature. This is not trivial, as the search must be undertaken in 4 dimensions across several domains, from the sun itself to the end of the solar system, following the phenomena as they propagate, taking care possibly of changing coordinate systems. Often, instruments are part of an observatory, which has a specific position in the solar system at a specific time. Whether, where and when to look is also not a trivial task.

In the third step, after the instruments have been found, heliophysicists will want to review the availability of suitable observations. They will determine whether suitable instruments were positioned at the relevant locations, determine the types of observations required (time series, images, spectra, etc.). They will want to determine whether instrument was making observations, and if yes, whether the coverage and quality of observations were adequate.

In the fourth step, once the suited observations are selected, heliophysicists will want to locate the data on the Internet, select, and eventually retrieve the required observations. For all domains, the system must know which types of data are held at what place and how to access this data no matter how it is stored (access protocols & formats). Scientists might want optionally to process selected observations (e.g. extract and calibrate), and possibly return data in differents format.

Once the data sets are downloaded to a local machine, heliophysiscists will use their own software to actually analyse, and eventually interpret the data.

HELIO wants to exploit the generic aspect of the basic problem, to reuse software components over and over again. Therefore, the HELIO architecture actually "translates" the generic problem into a technical infrastructure able to actually solve the basic problem. It extends the solutions to further variants of the problem. This is visualized in Figure 1. The four basic steps are translated into four categories of services: the metadata services, the discovery services, the data access services, and the enabling services. These services are provided either by the HELIO consortium or by independent external bodies.

Metadata services cover access to different kinds of catalogues. Of particular importance is the access to catalogues of events and features occurring on the sun and in the heliosphere. Metadata services include the Heliophysics Event Catalogue (HEC), the Heliophysics Feature Catalogue (HFC), the Data Evaluation Service (DES), and the Context Service (CXS). The HEC offers a centralised catalogue to query several dozens of solar and heliophysics event lists. The HFC applies image recognition algorithms to a collection of observations in order to extract features of the sun. A catalogue describes the behaviour of the features in space and time.

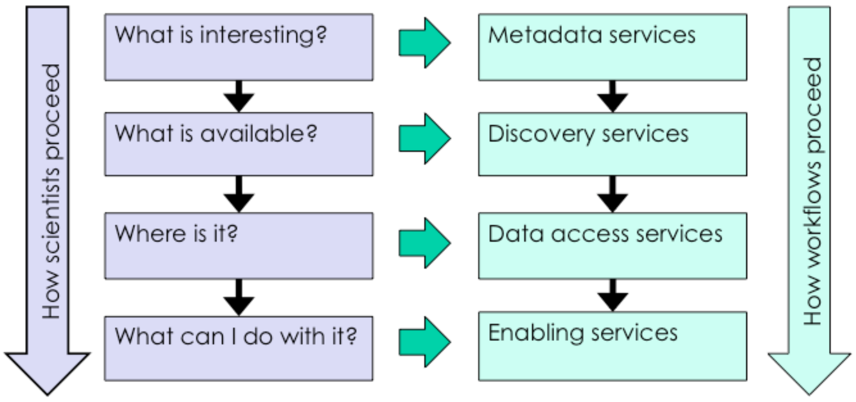


Figure 1: Mapping of the scientists view of the problem to the high level technological view of the system

Discovery services provide access to information allowing to explore what instrument has observed a specific phenomenon. They take into account the position of the observatory (e.g. spacecraft moving through the heliosphere) as well as their operating periods (observatories do not operate forerver). Services in this category include the Instrument Capabilities Service (ICS), the Instrument Location Service (ILS), and the Observation Coverage Service (OCS).

The data access services provide access to observations or raw data. It consists of a single service called Data Provider Access Service (DPAS), which enables unified access to the actual observation data.

Enabling services enhance the system by a multitude of advanced capabilities. Most important in this category are services allowing to store data as well as to process them. For this, the HELIO Processing Service and its companion the HELIO Storage Service implement access to a high-performance computing grid, e.g. for image recognition algorithms. Associated with them are services for authentication and authorization. A coordinate transformation service enables conversion between a couple of 2D-, 3D- and 4D-coordinate systems used in the solar and heliophysics domain. A propagation model service tracks propagation of an event through space and time. A semantic mapping service compares data with a different structure by using an ontology for mapping it to a global model.

The core services of HELIO are listed in Table 1 (Appendix A: HELIO Services). Further details can be found in the HELIO Concepts Document [2]. The implementation of these services will be discussed in Section 3.1 (Service Provider components) as well as in section 5 (HELIO Components).

Having described independent services, one particularly important point to note is that they need to work in concert. In Figure 1 this is shown by the black arrows. HELIO offers several ways to combine individual services into potentially complex tasks.

While many services provide their own standalone graphical user interface, the preferred access for the novice user will be the HELIO Front End (HFE), a browser-based user interface intended to solve the most common tasks. For advanced users HELIO provides an IDL API such that information from the system can directly be used in custom-made IDL programs. For expert users, HELIO offers programmatic access. A small Java library, called HELIO Client API, simplifies this access by providing client stubs for services and shields the user from configuration, security, discovery and failover issues.

Finally, HELIO services can also be handled through the Taverna workflow tool. Taverna is widely used in scientific applications. It enables orchestration of several services into more complex workflows. This is particularly useful for repetitive execution of a task. Taverna is used in two different flavours in the system, either through an instance of the Taverna workbench where users can design and implement their own workflows on any computer, or through execution of pre-defined workflows in Taverna server.

3 Structural architecture

This section outlines the structural or static view of the conceptual architecture. It shows the components involved in the overall system and how they are connected.

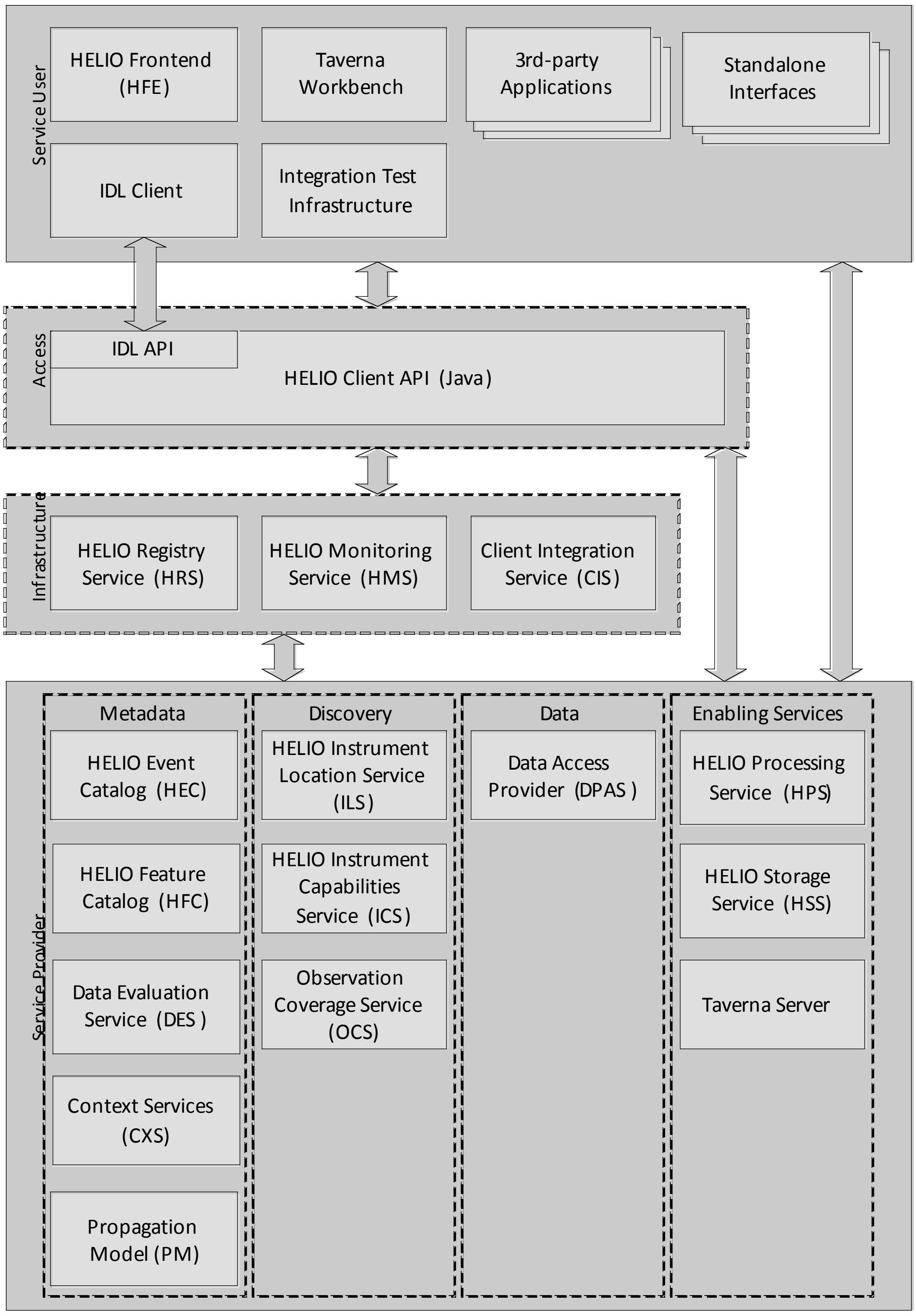

Service User
HELIO Frontend (HFE)
Taverna Workbench
3rd-party Applications
Standalone Interfaces
IDL Client
Integration Test Infrastructure
Access
IDL API
HELIO Client API (Java)
Infrastructure
HELIO Registry Service (HRS)
HELIO Monitoring Service (HMS)
Client Integration Service (CIS)
Service Provider
Metadata
HELIO Event Catalog (HEC)
HELIO Feature Catalog (HFC)
Data Evaluation Service (DES)
Context Services (CXS)
Propagation Model (PM)
Discovery
HELIO Instrument Location Service (ILS)
HELIO Instrument Capabilities Service (ICS)
Observation Coverage Service (OCS)
Data
Data Access Provider (DPAS)
Enabling Services
HELIO Processing Service (HPS)
HELIO Storage Service (HSS)
Taverna Server

Figure 2: Structural view of the main components involved in the HELIO infrastructure. The arrows denote communication flows

The design of the architecture is based on the core concepts of a Service Oriented Architecture (SOA) and on the mapping of the scientists view to the technical view introduced in the previous section (Figure 1).

Figure 2 shows the structural architecture of the components involved in the HELIO infrastructure. The diagram is divided into four main areas. At the bottom of the diagram you find the service provider components. They stand for the metadata, discovery, data and enabling services introduced above. While these services provide all scientifically valuable information to HELIO, the infrastructure components provide the information required to run the HELIO infrastructure. This includes management, maintenance and security components. The access layer components are connected to the underlying service provider and infrastructure components and integrate them as a whole. Through the combination of services new functionality can be provided such data filtering, data aggregation, or on a technical level failover handing. On top of the access layer components are situated the service user components. Service user components provide the interface between the infrastructure and the end users. End users can be human beings or computer programs. Service users can either access the service provider or infrastructure directly or through the access layer.

In the following sections the components are described in more detail.

3.1 Service Provider components

The service provider components are divided into four categories: metadata, discovery, data and enabling services.

3.1.1 Metadata Service Providers

In the first step of the generic process to find observation data the user wants to find features and events that are of interest to him or her. One may look for events with similar properties or features that could be connected somehow. This information is stored in the metadata catalogues.

The content of the metadata catalogues is compiled by the HELIO consortium. There are two kinds of metadata catalogues: Static catalogues and on-demand catalogues.

Static catalogues consist of a collection of tables stored in a relational database. Some of these catalogues are manually populated, others by a regularly executed script.

To access static catalogues, these services implement the common HELIO Query Interface (HQI). The HQI provides a set of methods to submit a time-range, a set of coordinates or a free-form query to the service. As query language the HQI uses the Parameterize Query Language (PQL) as standard defined by the International Virtual Observatory Alliance (IVOA). PQL expressions are valid URIs. Therefore PQL is optimal for REST-interfaces. Currently, there are on-going discussions if we should allow SQL-queries through the SOAP-interface. The returned data tables are rendered in the VOTable format. The HQI is further detailed in section 5.6.2 (The HELIO Query Interface (HQI)) and [6] and [7].

To date, HELIO provides two static metadata catalogues: the HELIO Event Catalogue (HEC) and the HELIO Feature Catalogue (HFC).

The **HELIO Event Catalogue (HEC)** provides access to various event lists. Events are observed on the sun as well as in the heliosphere. Their content, except for a few exceptions mentioned below, is provided by third-party institutions. The HEC regularly fetches and parses these lists and homogenizes the data to fit into a common data model. Table columns with similar content are named the same and time and coordinate values are transformed to the same format.

The HEC currently implements almost 50 lists. For better management it provides an additional list of lists called HEC_catalogue. HEC_catalogue contains the name, description and status of all implemented lists as well as a categorization of the lists. Categorization parameters are the type of the event (Coronal Mass Ejection, Flare, Solar Wind, Particle), the location where the events occurred (Solar, Interplanetary space, Geo, Planet) and the type of the observation (in-situ or remote). Access to the HEC is provided through the HQI.

The **HELIO Feature Catalogue (HFC)** stores information related to features of the heliosphere. While an event describes something that happened, a feature rather describes how something looks like. A good example for a feature is the shape of a sunspot and the description how it evolves over time. Usually, features are detected by image analysis algorithms. The HFC holds information derived by using feature recognition codes on images, etc. at various wavelengths. The features currently included in the catalogue are sunspots, active regions, filaments and coronal holes identified in solar images and type II and III radio bursts detected in time/frequency plots.

The HFC contains three different types/levels of information:

- Level I data includes the information derived from the image processing routines describing the location and outline of features.
- To produce the level II data, features detected in the level I data are tracked and the catalogue can be used to determine if a feature seen in one image is the same as the feature seen in subsequent image.
- The level III catalogue is derived from the level II data and shows where things have happened. For example, if a segment of filament disappears or a group of sunspots emerges. This catalogue is more analogous to an event catalogue and some of the data in the level III data is similar to lists in the HEC. This data is made available to the end user through the HEC.

The HFC can be used as standalone tool. Parts of it are integrated into the HELIO Frontend. More information about the HFC can be found in [1].

As complement to the static catalogues HELIO provides some **on-demand catalogues**. Rather than to store all information in tables the answer to queries sent to these kind of services need to be computed at runtime. The answer can be tabular data in the VOTable format or it can contain images or plots. HELIO provides three on-demand metadata services: the Data Evaluation Service (DES), the Context Service CXS and the yet embryonic Propagation Model (PM).

The **Data Evaluation Service (DES)** acts as interface to the Automated Multi Dataset Analysis (AMDA) infrastructure [18]. AMDA provides a collection of tools to access and analyse space physics data. AMDA is a sophisticated data analysis tool with its own web interface. The DES simplifies AMDA by providing a set of predefined data-analysis functions tailored for the needs of HELIO. The functions ask for a set of parameters that are relayed to AMDA for evaluation. The result is formatted as VOTable. As query language DES implements an extended version of PQL. This has the advantage that DES can be accessed through the HQI. As DES is still in the construction phase no official document has been released yet.

The **Context Service (CXS)** gives access to predefined plotting services for instance to create a timeline plot of solar activity for a given date range. Presently the CXS bases on Astrogrid's Common Execution Architecture (CEA). AstroGrid CEA has some rudimentary online documentation [21] and some more verbose information through access to an installed CEA instance (e.g http://msslkz.mssl.ucl.ac.uk/helio-cea/doc/index.jsp). CEA can be connected to any (a) command line application, (b) an existing web form based HTTP application or (c) a custom written Java application. In HELIO we mainly use (c) to run IDL scripts.

Currently the CXS generates plots of GOES light-curves, solar images over-plotted with the location of flares and a plot of the parker spiral. All plots are generated on demand.

The **Propagation Model (PM)** simulates the propagation of the effects of a solar event through space and time. This allows to relate observations made at different locations and time to the same event. A prototype specification can be found in [14]. An embryonic implementation is currently being tested.

The outcome of the metadata services is generally a set of time ranges and possibly locations of interest.

3.1.2 Discovery Service Providers

The next step in the process of locating interesting data is first to find out which instruments are of importance to a scientific question, second, if these instruments have been in the right place to observe something and, finally, if they have made any observation that could be of help. To address these questions HELIO provides the Instrument Capabilities Service, the Instrument Location Service and the Unified Observation Catalogue.

All discovery services are implemented as static catalogue services providing the HQI for access and PQL as query language. Return values are rendered as VOTable.

The **Instrument Capabilities Service (ICS)** describes more than 250 instruments and holds information about the characteristics of the associated observatories. This description includes the type of instrument xxx. The ICS is used to reduce the amount of instruments to query for. The content of the ICS is populated manually.

Note that not all instruments described in the ICS are available through HELIO. The list of actually available instruments can be retrieved through the Data Provider Access Service (DPAS, see below).

The **Instrument Location Service (ILS)** contains details of the locations of all main planetary objects in the solar system and of spacecraft that have travelled on a complex trajectory. The information is provided with a cadence of one value per day. To date, the ILS provides trajectory data for all eight planets and the following spacecraft: xxx. The core information stored in the ILS is date and position.

The **ILS** also contains a table of the dates of Key Events in the life of several spacecraft including launch, planetary flybys and insertion into orbit around the target object.

The **Unified Observing Catalogue** is used in two different ways and the types of information associated with these are quite different:

i. For (remote-sensing) instruments that observe the Sun with restricted fields-of-view, the UOC contains information about the observations and make it possible to filter data requests based on whether the instrument was observing the required target. This can be used as a filter before the DPAS is being accessed.

ii. Where there are problems accessing the observations from some instruments – either because not all of them are on-line or because of complexities in the archive interface – the UOC contains the details of what is available and the URLs that should be used to access the data.

Entries in the UOC in the first category are used as part of the search process. Those in the second category are used by the DPAS in order to support access to certain instruments – that the DPAS should use the UOC is specified in the Provider Access Table (PAT).

The basic outcome of the discovery services is a list of instruments and potentially a narrowed date range that are used for further queries.

3.1.3 Data Service Providers

Using the outcome of the metadata and discovery services the user now wants to query for real data. This is done through the Data Provider Access Service (DPAS), which provides uniform access to a multitude of archives with data about heliospheric observations. The DPAS implements connectors to various types of archives, such as FTP- and HTTP-archives, web services, relational databases and other virtual observatories. The DPAS requirements are defined in [5].

The protocol of the DPAS follows the HQI. Indeed you can issue PQL queries for data. The found files are returned as a list of URLs in a VOTable.

To date the DPAS supports access to more than 150 instruments from more than 40 observatories. The actually available instruments are available through the Provider Access Table (PAT).

3.1.4 Enabling Service Providers

The processing category holds services for on-demand processing of data or meta-data. Depending on the scientific question asked to HELIO some information cannot be prepared in advance but has to be computed based on given parameters. HELIO provides several types of processing components:

Taverna [15] is a workflow engine suited to combine multiple services into a more complex workflow. Taverna is an external project with close relations to HELIO. **Taverna Server** is a subcomponent of Taverna that allows to run workflows in a server environment. HELIO intends to exploit Taverna Server to run user defined workflows that solve specific tasks. More information about Taverna server can be found at in section 5.2 (HELIO Workflows).

The **HELIO Processing Service (HPS)** provides access to a high-performance computing infrastructure located at Trinity College Dublin, Ireland. The infrastructure is based on the gLite grod toolkit. The HPS is used for resource intensive data processing such as image analysis. In a first step HELIO is going to use the HPS to run selected, what we call pre-canned functions. I na second step HELIO may become able to run any job. Further details about the HPS can be found in [4].

The **HELIO Storage Service (HSS)** acts as utility service for the HPS and other services to store large result sets. It relies on storage facilities at Trinity College Dublin. Further details about the HSS can be found at [4].

3.2 Infrastructure components

The infrastructure category provides utility services for the management of the HELIO infrastructure. These components are usually transparent to the end user and do not provide any information of scientific value.

The **HELIO Registry Service (HRS)** is a directory service for service discovery. All instances of a service are registered in the HRS. Clients can query the HRS to find these instances. In addition the HRS registries provides specifications on how to access a services. This includes definition of the accepted service parameters and a formal description of the returned result. The HRS is based on AstroGrid's registry service. More information can be found in section 24 (HELIO Registry Service (HRS)).

The **HELIO Monitoring Service (HMS)** monitors the system by frequently polling the status of the deployed service instances. The list of deployed services instances is retrieved form the HRM. In combination with the HRS it provides fail over and load balancing capabilities to the HELIO infrastructure. Moreover, it provides rudimentary profiling capabilities by gathering statistics about the response time of the different services.

The **Client Interaction Service (CIS)** implements the basement for authentication in HELIO. It manages user profiles in a central place. The CIS is further described in [4].

3.3 Access layer components

Depending on their needs client applications may choose to directly access individual HELIO services or they may use the Java-based HELIO Client API. The HELIO Client API facilitates access to the system by shielding users from the underlying infrastructure. It offers:

- Transparent discovery of services.
- Rudimentary load balancing and fail over.
- Automatic handling of security and user profile management.
- Client stubs to access different service providers in a uniform way.
- Utilities to combine services to solve more complex tasks.

The full specification of the API can be found in [9].

3.4 Service User components

The service user category contains components to access HELIO for users. In this context a user can be a human beings as well as a computer program.

The **HELIO Front-end (HFE)** provides browser-based access to HELIO both for the public and for scientists. It allows the user to perform the most frequent use cases in a user friendly way. The HELIO Front End is further described in [10].

The **Taverna Workbench** allows users to define custom workflows for their own specific scientific use cases. Taverna Workbench features a visual composition and configuration environment for workflows. It does not put any explicit requirement of programming experience on the end user.

The **HELIO IDL client** enables access to HELIO through the Interactive Data Language (IDL) scripting environment. With IDL, users can interactively communicate with the HELIO system and, this way combine the HELIO capabilities with advanced data analysis tasks that might run only in IDL. Furthermore, they may write complex scripts in the de facto standard data analysis language of the space science domain.

The **3rd-party Applications** component summarizes all possible external client applications. This includes other virtual observatories or data warehouses and existing or future graphical client applications that integrate HELIO.

The **Integration Test** Infrastructure is frequently executed to run a set of integration tests. It is described in appendix A of the Test Specification document as well in appendix B of the HELIO API document [9].

Most HELIO services come with their own standalone interface allowing them to be directly used over the web without integration with any other system. They are heavily used for data validation and testing.

4 Behavioural architecture

The behavioural or dynamic architecture describes how selected components outlined in the previous section work together.

4.1 Information Flows

Figure 3 shows the main information flows within the HELIO infrastructure.

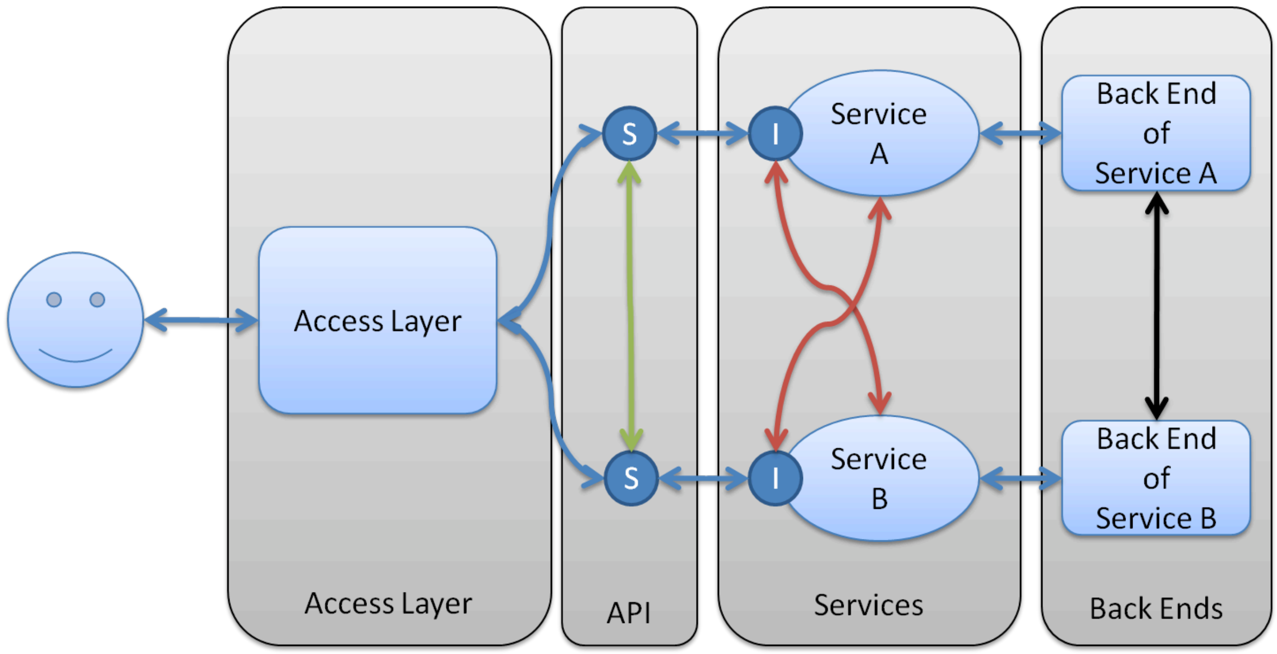


Figure 3: Information flows in HELIO

The ***service stubs*** (represented in Figure 3 by a circle labelled s) represent elements that interface the clients to the services and the logic that handles the information flows that are pertinent to the Access Layer (such as authentication) but are not of direct interest to the user.

The interfaces implemented by the service stubs are part of the HELIO API layer, described in the HELIO API document.

In this abstraction HELIO comprises the following information flows.

- The information flows that connect some of the back ends.
- The information flows that connect the HELIO services to the Back Ends. These information flows can be:
    - Ingestion mechanisms where scientific data is copied (and maybe processed) to databases local to the services for further processing or querying;
    - Access, where the service accesses data from the back ends;
    - Control information, where services control the back ends;
- The information flows that directly connect different services to each other;
- The information flows that connect the stubs to the services within the HELIO API;

- The information flows that connect the different stubs;
- The information flows that connect the access layer to the stubs;
- The information flows that connect the user to the access layer.

4.2 Design

The simple abstraction of section **Error! Reference source not found.** does not explicitly include the workflow engines and special services such as the "Registry and Monitoring Services" that are used to discover other services and data and are, therefore, of particular importance and of special nature with respect to the other services. A more complete abstraction of the HELIO architecture that explicitly includes these items is represented in Figure 4.

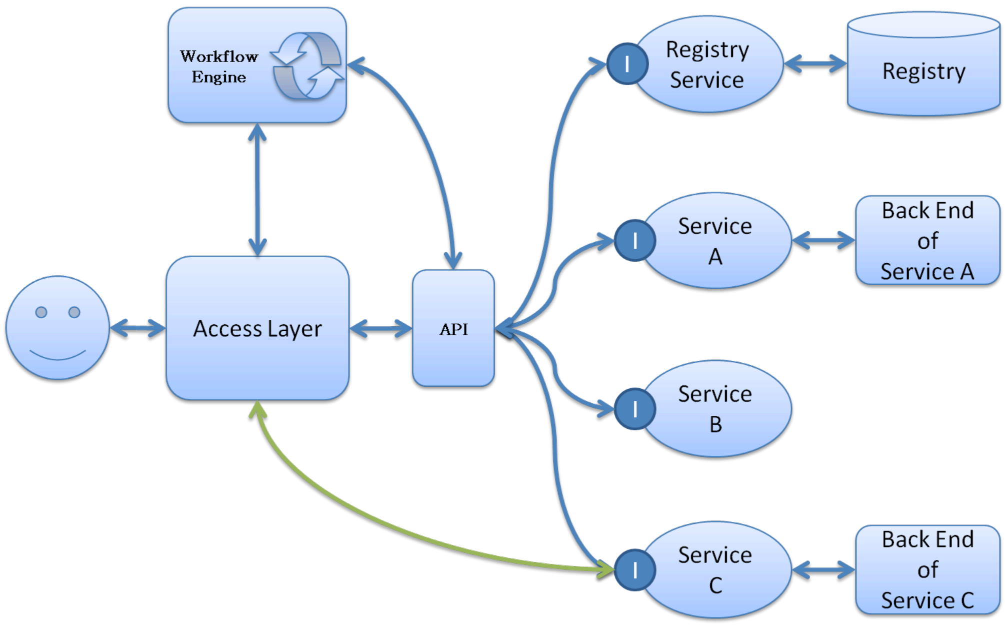


Figure 4: Complete abstraction of the HELIO architecture

Figure 4 includes:

- A workflow engine;
- A Registry Service that stores all the information describing the instances of the HELIO services;
- The database of the registry service;
- The HELIO API that comprises the stubs to the services.

As the interactions between the Access Layer, the workflow engines and the API is complex and fundamental to HELIO, it is described in detail in Figure 5.

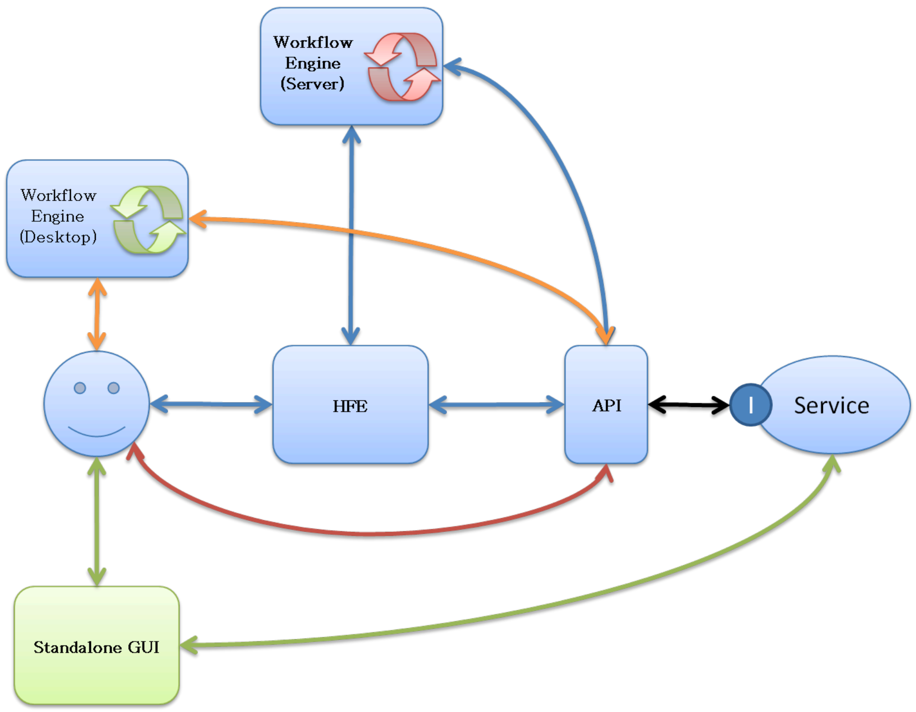


Figure 5: Details of the Access Layer

Figure 5 decomposes the Access Layer that contains the following:

- The HELIO Front End (HFE) and the "TAVERNA as a service" workflow engine;
- A desktop instance of TAVERNA (or another workflow engine such as KEPLER);
- Other standalone interfaces offered by some of the services;
- Direct access to services by the user or a program written by the user through the API.

To cater for the different HELIO users, there are different paths and modalities with which the HELIO user can access the services:

- Through the HELIO Front End (HFE), the path described with the blue arrows in Figure. In this case, the user logs into the HFE and, from there, accesses the services either independently or orchestrates them by executing pre-defined workflows in the TAVERNA server.
- Direct invocation of the API, the path described with the red arrow.
- Through an instance of the TAVERNA desktop local to the user, where workflows can be designed, experimented and executed.
- Using the standalone interfaces that some of the services offer.

4.3 Advantages of the design

4.3.1 Duality of HELIO Services

The presence of the HELIO API, the HELIO Front End and the standalone Graphical User Interfaces allow the dual invocation of the HELIO services. HELIO services can be invoked both from the workflow engine and as standalone services as the access layer does not impose any constraint on the HELIO API.

4.3.2 Workflow Agnosticism

The "workflow agnosticism" of HELIO poses the following constraints in the design:

- The registry of services must not be specific to Taverna;
- Handling of authentication information must not be specific to Taverna.

Because nothing specific in the architecture is tied to the TAVERNA workflow engine, other workflow engines should, in principle, be supported by the architecture as well as user-defined programs and scripts.

4.3.3 Flexible deployment

Services in HELIO should be able to be co-located in order to allow for different kinds of optimization. In HELIO there will be two main kinds of optimization:

- Location of two or more services within the same Web Service container to reduce the time needed for the information exchange within the services;
- Location of two different data sets in the same local database to optimize searches in the joined space of the two data sets.

Each item of this location policy has different consequences on the design:

- Location and co-location policies within the same containers have consequences on how the registry stores and queries the information on the services. In fact the information on the location of the service becomes important to select among different instances of the same services;
- Location and co-location policies of data sets are reflected in the ingestion mechanisms and query interfaces.

The architecture, via the location of the services through the HELIO Registry Service allows for optimization based on locating and co-locating services.

4.3.4 Policy Compliance

Authentication (performed by the Community Interaction Service - CIS) and Authorization (performed locally in the services that need it) will ensure that the policies of the back ends will be complied with.

4.3.5 Need to know basis

The architecture ensures that the user must really be concerned only with the detail he/she is interested in.

5 HELIO Components

In this section the components involved in the architecture are further detailed. This section is constantly updated.

5.1 The HELIO Front End

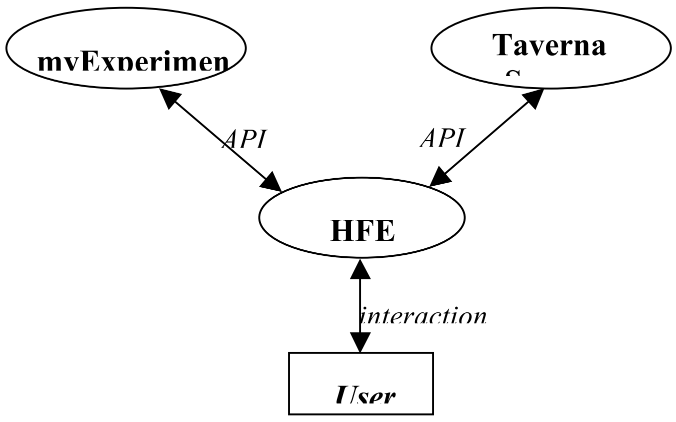
The HELIO Front End (HFE) is the centralized portal to the HELIO system, a visually appealing place where the user interacts with HELIO. It provides access to the most common functions of the platform as well as a user space for them to be able to save a history of their work.

The interface design in the HFE is a very active issue; it must provide the flexibility and modularity required for any of the workflows to be added automatically.

Currently there is a team at the FHNW[1] working on a specialized interface so the user will feel comfortable and will not be overwhelmed by an overly complex interface.

5.2 HELIO Workflows

Detailed information about workflows can be found in [11] so only broad concepts will be included in this document. Taverna is an open source workflow system that is actively developed and supported by the MyGrid team.



HELIO workflows will be constructed using the Taverna workbench [15] as a GUI (5.3). After they are finished and tested they will be uploaded to the myExperiment [16] repository and associated with a specific myExperiment user group. There they are available for scientist to download, use, and modify. Workflows belonging to a specified group can be made available also via the HFE. The myExperiment API can be used to enable this connection. The execution of these workflows can be scheduled on the Taverna server (5.4). The Taverna Server again provides an API for that task.

5.2.1 Workflow uses in HELIO

Workflows can have different purposes.

a) Scientist can test their theories

[1] University of Applied Sciences North-western Switzerland

b) Application of scientific methods on large amount of data

c) Testing of services

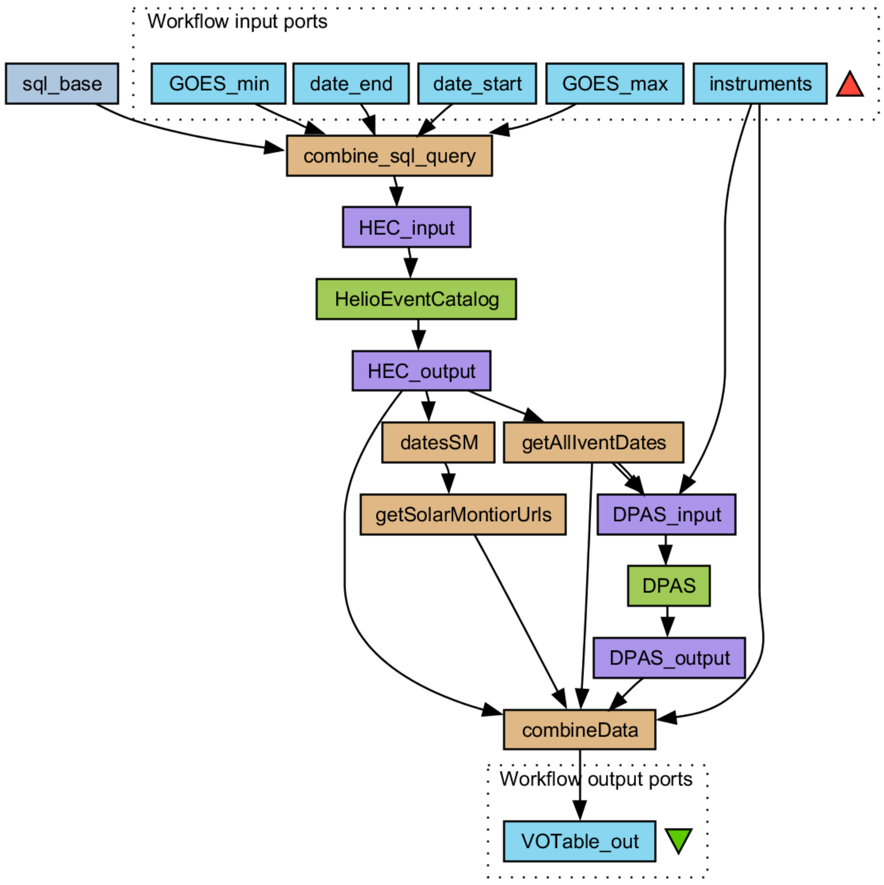


Figure 6: Example of a HELIO workflow

Figure 6 shows an example of a HELIO workflow that looks up events from the HELIO Event Catalogue (HEC)[2] for a specific time interval and filters those where all the specified instruments have made measurements.

URLs of images from the Sun at the peak time of the event are added from the Solar Monitor service to give some context information about the event. All data is then combined into one output file in VOTable format.

5.3 The Taverna Workbench

Taverna Workbench is the graphical interface to create workflows. All HELIO services are published as standard web services. These web services can be used to create more complex tasks by combining their functionality or modifying their outputs. It is a powerful tool for scientists to explore the functionality of HELIO and third party services. Taverna can work with both SOAP and REST web services, but since SOAP services are well defined, working with them is easier. Taverna Workbench is keeping the provenance of workflow runs. Scientists can check at all stages how a service was called and exactly what it returned. Scripts between services can perform additional operations on the data.

[2] See HELIO Concepts Document [2]

Taverna Workbench is independent of the HELIO Architecture. Workflows created with it can be seen as providing additional 'complex' services, if they are being executed from the HFE.

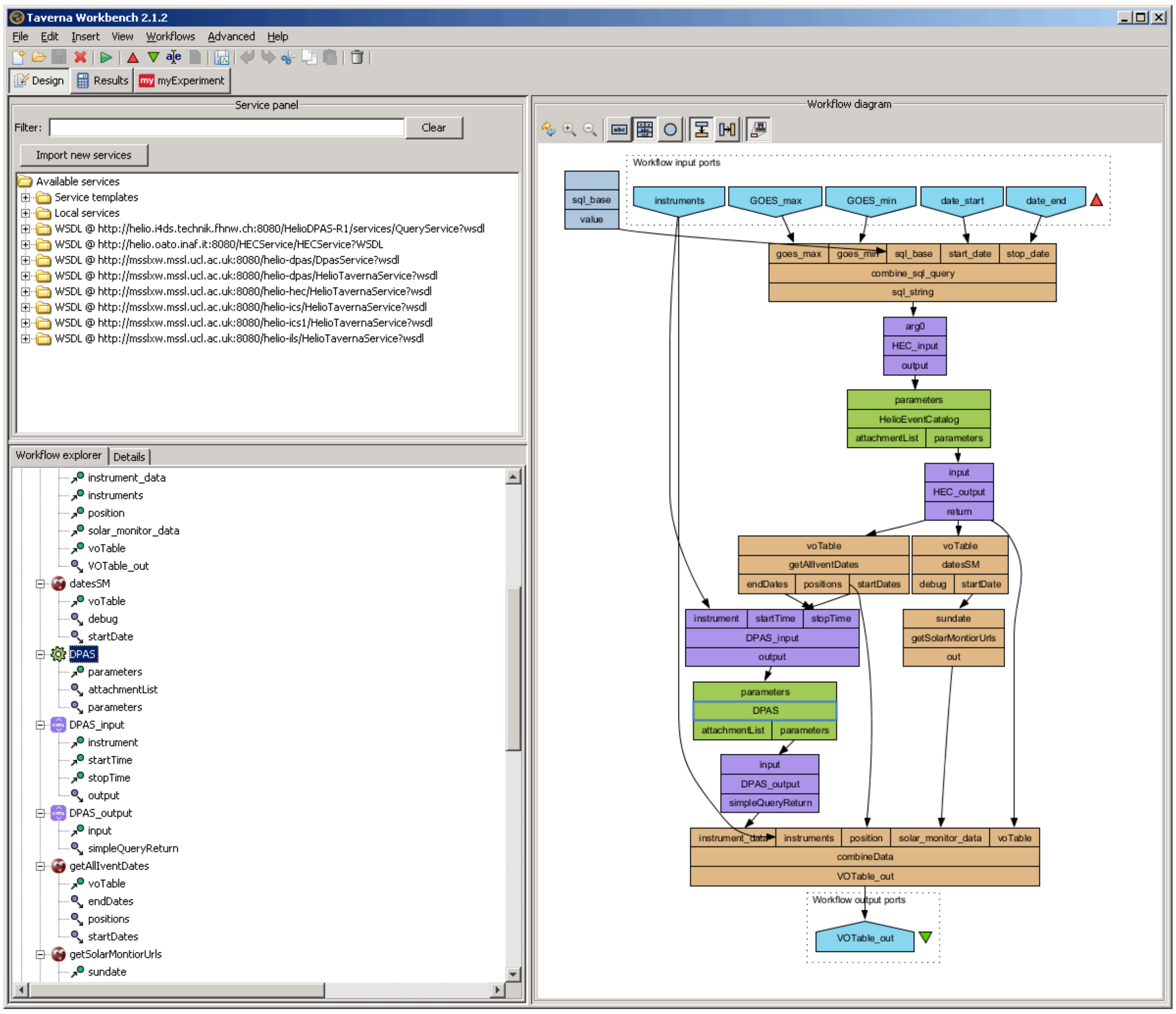


Figure 7: Taverna Workbench GUI

5.4 The Taverna Workflow Engine Server

Taverna Server is a webservice interface (developed mainly in HELIO) to a multi-user workflow engine that is able to execute workflows as defined by the Taverna Workbench. It supports both SOAP and RESTful access interfaces, and allows users of the service (or suitable portals, such as the HELIO Front End) to provide execution of workflows without having to integrate the Taverna workflow engine directly. It also ensures that different workflows and different users are strongly separated from one another, so that one user does not adversely impact another, and provides the capability to users to have notifications delivered when key events happen to their workflow runs (particularly useful with a long-running workflow).

Internally, Taverna Server is a dual JAX-WS/JAX-RS service implemented within the Apache CXF and Spring frameworks, and incorporating a copy of the Taverna Command-line Tool. It works through exposing a series of common models for access to and management of workflows so that the same capabilities are made available through all interfaces. The notification fabric allows the delivery of an event to multiple consumers, notably including the sending of email, SMS or instant message, the update of a Twitter status, or the addition of a message to an Atom news feed. The common management interface allows the whole server to be managed from both a web interface and also by the industry standard JMX system (enabling direct management by a great many Java tools). The security aspects of the server are handled through the use of Spring Security (a series of plug-ins for Spring) that enable the application of access control policies at the level of individual resources and operations; no access to any workflow is possible unless it is explicitly granted by the security framework.

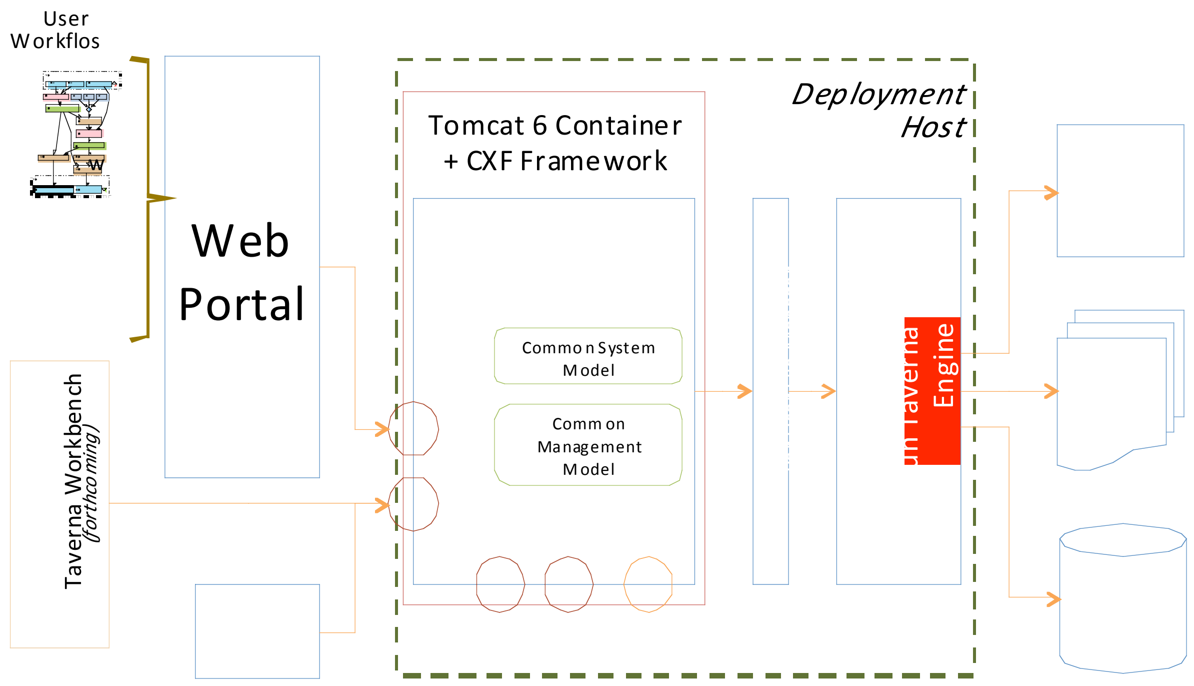


Figure 8: High-level Architecture of Taverna Server

The high-level architecture of Taverna Server is shown in Figure 8, which shows how the server framework fits with respect to a range of clients (including Web Portals), how the workflow engine may access many different types of back-end service as part of processing a workflow (particularly including web services), and how internally it is composed of a webapp that delegates accesses to a per-user file manager engine and a per-workflow execution engine. Internally the webapp is structured as a collection of "beans" (simple Java components) that implement a small self-contained piece of functionality (the Spring framework mentioned in the preceding paragraph is responsible for the management of these beans, their lifecycles, etc). The core of the system of beans is illustrated in Figure 9 though more exist that are fundamentally less interesting (e.g., for the correct translation of exceptions into web faults).

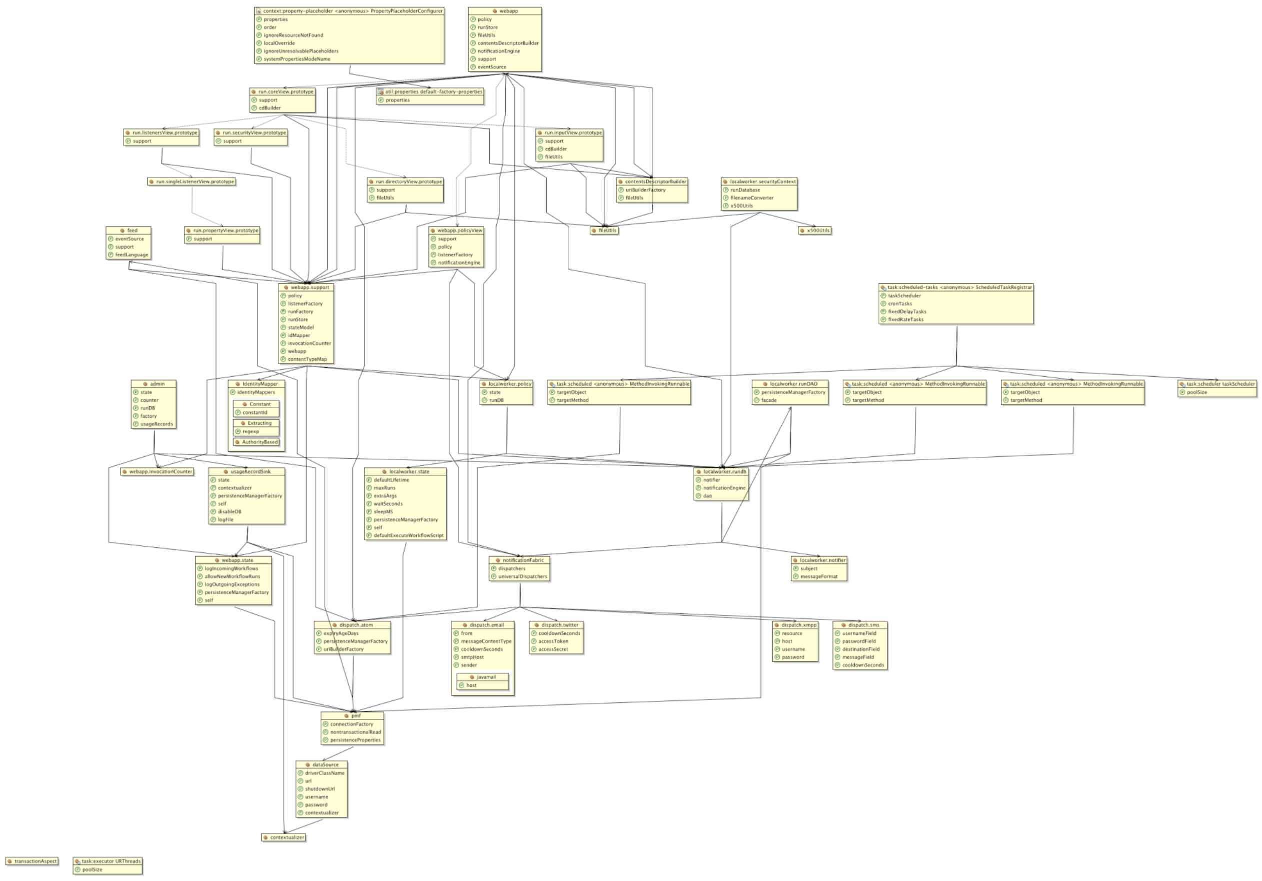


Figure 9: Taverna Server internal bean-level architecture

5.5 The HELIO API

Between the access layer and the service there is an HELIO API that shields from the user the details of the connection technology to the remote services and the information flows that are not of direct interest to the user (logging, security, etc...). The HELIO API is composed of a set of stubs that connect it to the different services, and the logic that connects the stubs to each other. The HELIO API is detailed in the HELIO API document [9].

5.6 The HELIO Services

Although HELIO services offer different functionalities, they are all based on the same generic architecture.

5.6.1 Generic architecture of the Service

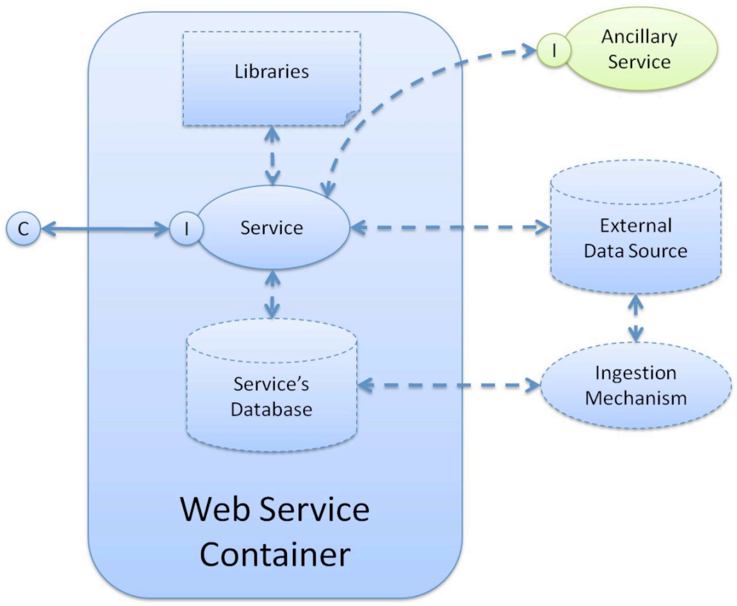


Figure 10: Generic architecture of a HELIO service

Each service **may** be comprised of one or more components (not all the services need to comprise the same components and not all services need to have the same connections). The components are:

1. A client (present in Figure 5 as a circle labelled as "c") that represents either a TAVERNA or standalone GUI instance or the client code of the HELIO API;
2. An interface (present in Figure 5 as a circle labelled as "i") that represents the service's interface;
3. The service itself: the code that implements the functionalities of the service;
4. The service's database: the database used by the service;
5. An ingestion mechanism: the mechanism with which the service database is populated (it can be by hand, web form interface or harvesting daemons);
6. Ancillary Service(s): other HELIO services that is invoked directly from the service;
7. External Data Source(s): the remote services that provide data to the service;
8. Libraries: code common to several HELIO services.

5.6.2 The HELIO Query Interface (HQI)

The HELIO Query Interface gives a client a common interface to query data. All implementations of the HQI expose a Service that conform both to a SOAP and REST style using the HTTP protocol. The HQI defines the response to be a VOTable based on IVOA specifications. The query language is an extension of the IVOA Parameter Query Language (PQL). PQL is a simplified query language and is best suited when the data model is well defined. HELIO works mostly in a very time dependent domain so some optional extensions were made to PQL for using date and time facilities.

Design

The architecture of the HELIO Query Interface comprises the following components:

1. The handler that handles all requests.
2. VO-Table creator: a library to create or generate VOTables.
3. HELIO Service: the implementation of the service that exposes the HELIO Query Interface.
4. Database: the database, if any, of the service.

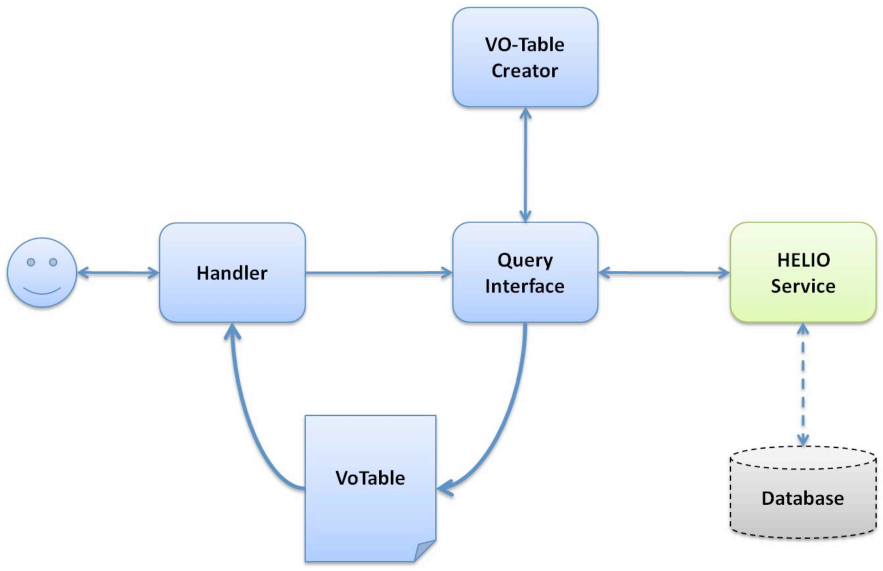


Figure 11: HELIO Query Interface

Relational Database

The first component developed by HELIO to conform to the HQI was targeted at relational databases. The HQI is generic enough to work on any relational database that uses a jdbc (java database connectivity) driver, which most common databases supply. The administrator through a properties file can setup custom queries, if needed, as well as customizing the VOTable to conform to the data model, such as setting particular utypes.

Though the HQI component can be used on any relational database, certain particular services that use the HQI are mentioned below (more details of these services can be found in the Concepts Document [2]):

- HEC – HELIO Event Catalogue - consisting of over 40 tables of particular HELIO events.
- UOC – Unified Observing Cataloge – A Catalogue that has harvested detailed observations of various instruments and placed into a relational database to be queried.
- ICS – Instrument Capability Service – Detailed information for an instrument i.e. in-situ, remote, date in service.
- ILS – Instrument Location Service – Location of a particular instrument.

- MDES – MetaData Evaluation Service --

HQI External: Data Provider Access Service (DPAS)

HelioPhysics community already has many services, ftp and http sites that have access to the actual data (normally contained in FITS or CDF) files. The second component to conform to the HQI interface is the DPAS service, which acts as a façade layer to gain access to the data of many external services. This allows the Helio clients and the API to use a common interface to access many providers and catalogues instead of having custom code for each particular provider.

DPAS gains access to two main Providers of VSO and CDAWEB along with several http and ftp sites for various instrument data. UKSSDC services will be accessible through the DPAS by September 10th 2011.

5.6.3 HELIO Long Query Interface (HLQI)

The HQI interface is always a synchronous query that will perform the query each time the service is used. The HLQI was developed to allow longer queries and most importantly to save the votable on the server. URL references to the saved Votable can then be passed further in a client's workflow or could be used many days later. The initial input query to the HLQI is exactly like the HQI, the difference though is a unique ID number that can be used to poll when the query is complete and to gain a URL reference to the resulting Votable. It also conforms to both a SOAP and REST style specification. ALL components that implement the HQI interface, also implements the HLQI interface.

The procedures a client would do for calling a service that implements the HLQI:

- Query Service, gather unique ID.
- Poll Service with Unique ID to get a Status.
- If Status is Complete then gather URL reference to resulting VOTable.

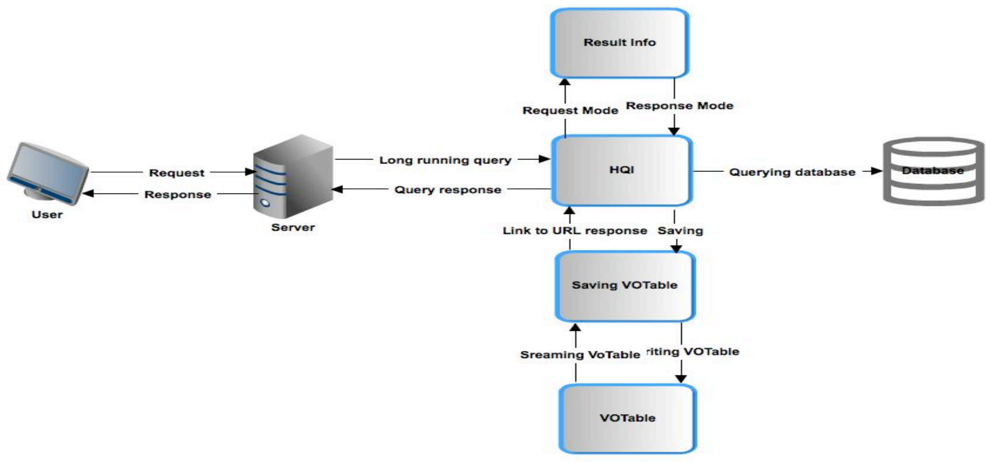


Figure 12: High level architecture of the HELIO Query Interface

5.6.4 Context Service (CXS):

The Context Service is a way of running light weight commandline applications on an external server. The applications are targeted at providing Helio Context information to help the user decide if particular observations are worth pursuing. The CXS conforms to the IVOA specification of 'Universal Worker Service' more details can be found here: Universal Worker Service Currently the CXS has three particular applications:

- GOES plotter
- Flare Plotter
- Parker Model Propagation Model
- UKSSDC has particular applications that would fit well into the CXS model and target to be implemented by Mid-October 2011.

5.6.5 HELIO Registry Service (HRS)

The HELIO Registry Service conforms to the IVOA specification: Registry Interface Specification Document. The HRS contains metadata about a particular 'Resource', a Resource could be a Service, CatalogService, or Metadata about a particular instrument. The Registry conforms to several XML schemas to populate metadata for these particular type of Resources. The HRS has two main goals for Helio architecture:

- Allow clients and API to find a location of a particular Service i.e. HQI service.
- Return more pertinent metadata to help the user understand the details of a particular Resource. This normally includes:
    - Detailed Column information to help construct Queries to an HQI service.
    - Input Parameters to a particular Context Service (CXS).
    - Possibly coverage information of a particular instrument.
    - Descriptions of a particular Provider and Instruments.

6 References

[1] Aboudarham, J. Bonin, X. Renié, C., Fuller, N., Perez, D.: **HELIO_OBSPM_R2_nn_RP_HFC_2.1: Heliospheric Feature Catalogue**

[2] Bentley, R.D. **HELIO-UCL-N1-004-TN: HELIO Concepts Document**

[3] Bentley, R.D. **HELIO-UCL-N1-005-PM: Release Strategy**

[4] Pierantoni, G. **HELIO-TCD-S3-001-SP: Processing and Storage Services.** http://helio-vo.eu/internal/Documents/Deliverables/HELIO-S3-001-d1_ProcessingStorage_100625.pdf

[5] Gangloff, M., **HELIO-UPST-N3-001-RQ: Review of HELIO Provider Requirements** http://helio-vo.eu/internal/Documents/Deliverables/ HELIO_UPST_N3_1_Provider_Requirements_v0.3.pdf

[6] Dalla, S., Benson, K., Shetty V., **Helio Simple Time Protocol**, Version 0.2, Working Draft. http://helio-vo.eu/internal/Documents/Project/HelioSimpleQuery_0.2.pdf

[7] Benson, K. Shetty, V., **Helio Query Interface, Definition of required capabilities,** Version 1.0, http://helio-vo.eu/internal/Documents/Project/Helio_FullQuery_0.2.pdf

[8] Marassi, A., Santin, A., Messerotti, M., **HELIO-INAF-S2-001-SP: Metadata Server Infrastructure HEC Architecture** http://www.helio-vo.eu/internal/Documents/Deliverables/HEC_Metadata_Server_HEC.pdf

[9] Soldati, M. **HELIO-FHNW-S5-001-SP: HELIO API** http://helio-vo.eu/internal/Documents/Deliverables/HELIO-S5-002-d2_API_100701.pdf

[10] Soldati, M. Guevara, D., Wyss, H.P., Csillaghy A., **HELIO-FHNW-R3-001-RP: Report on the implementation of the HELIO user interface** http://helio-vo.eu/internal/Documents/Deliverables/HELIO-R3-001-d1_Web_Frontend_1.0.pdf

[11] Le Blanc, A. **HELIO-UNIMAN-S4-001-TN: Taverna Workbench for HELIO**

[12] Csillaghy, A., Seidler, K., **HELIO-FHNW-S1-002-RP Infrastructure Monitoring and Profiling Tools Documentation** http://helio-vo.eu/internal/Documents/Deliverables/HELIO-S1-002-d2_Monitoring_Tools_1.0.pdf

[13] Fellows, D., Le Blanc, A. **HELIO_UNIMAN_S4_002_UM: Workflow Repository including Documentation**

[14] B. Lavraud, C. Jacquey, E. Budnik, M. Gangloff, Nicolas André, Vincent Génot, A. P. Rouillard, Baptiste Cecconi: **HELIO-UPST-R2-005-SP: Heliospheric propagation tool Specification** http://helio-vo.eu/internal/Documents/Deliverables/HELIO-R2-005-d3_PropagationModel_100530.pdf

[15] Taverna, http://www.taverna.org.uk.

[16] myExperiment, http://www.myexperiment.org

[17] R. Perrey and M. Lycett, “Service-oriented architecture,” in Applications and the Internet Workshops, 2003. Proceedings. 2003 Symposium on. IEEE, 2003, pp. 116–119.

[18] Jacquey, C., V.Génot, E.Budnik, R.Hitier, M.Bouchemit, M.Gangloff, A. Fedorov, B. Cecconi, N. André, B. Lavraud et al., “Amda, automated multi-dataset analysis: A web-based service provided by the CDPP,” The Cluster Active Archive, pp. 239–247, 2010.

[19] F. Ochsenbein, R. Williams, C. Davenhall, D. Durand, P. Fernique, R. Hanisch, D. Giaretta, T. McGlynn, A. Szalay, and A. Wicenec, "VOTable: Tabular data for the Virtual Observatory," in Toward an International Virtual Observatory, ser. ESO Astrophysics Symposia, P. Quinn and K. Górski, Eds. Springer Berlin / Heidelberg, 2004, vol. 30, pp. 118–123, 10.1007/10857598_18. [Online]. Available: http://dx.doi.org/10.1007/10857598_18

[20] C. Jacquey, V. Génot, E. Budnik, R. Hitier, M. Bouchemit, M. Gangloff, A. Fedorov, B. Cecconi, N. André, B. Lavraud et al., "Amda, automated multi-dataset analysis: A web-based service provided by the CDPP," The Cluster Active Archive, pp. 239–247, 2010.

[21] **Astrogrid Common Execution Environment (CEA) services.** http://www.astrogrid.org/maven/docs/HEAD/applications/index.html (last visited: 05-Sept-2011)

Appendix A: HELIO Services

Table 1 provides a list of HELIO services.

Name	Acronym	Description
Heliophysics Event Catalogue	HEC	Performs queries on event catalogues
Heliophysics Feature Catalogue	HFC	Performs queries on the feature catalogues
Metadata Event Service	MDES	Handles event's metadata
Context Service	CXS	Performs queries on the observations' context
Instrument Capability Service	ICS	Performs queries on the capabilities of the different instruments.
Instrument Location Service	ILS	Performs queries on the location of the different instruments.
Unified Observatory Catalogue	UOC	Helps to refine the queries of the DPAS
Observation Coverage Service	OCS	Performs queries on the coverage of the observations
Data Provider Access Service	DPAS	Allows to query data sources
HELIO Storage Service	HSS	Provides access to storage facilities
HELIO Processing Service	HPS	Provides access to processing facilities
Coordinate Transformation Service	CTS	Performs transformations between the different coordinate systems
Semantic Mapping Service	SMS	Maps Semantic Information
HELIO Registry Service	HRS	Performs queries on the available HELIO services
HELIO Monitoring Service	HMS	Monitors the other HELIO services.
Community Interaction Service	CIS	Provides Authentication

Table 1: HELIO services