4. Central Services

A trivial control system can be built of just two parts: a set of low-level equipment front-ends and a set of high-level user applications. Both sides talk to each other through some common protocol, so that the front-ends' data is reflected in the applications, as well as the user's actions are reflected in the equipment.

The larger the system grows, the less this simple model can satisfy its demands:

- There is no provision of data relevant to the entire system, such as a list of equipment.
- The number of possible interconnections increases exponentially, causing addressing, performance, and licensing issues.
- It becomes harder to maintain uniform implementations of the common communication protocol throughout the system, so applications may have to talk differently with different equipment.
- It becomes harder to add new features into all the front-ends, such as authentication or archiving mechanisms.
- It is difficult to implement a function that would involve a number of distributed front-ends at the same time, such as a transaction service.

In this document the term *Central Service* refers to a system supplying the common needs of both high-level applications and front-ends, and directly accessible through some sort of external calls. For each service we describe functional requirements, as well as essential features of its external interface. The latter should be applied to an API used by the clients, which include high-level user applications, front-ends, and other central services.

No.	Requirement	Source	Priority
CXR-CS-10	Each central service shall include a graphical user interface (GUI) for remote configuration and monitoring.	A. Petrov 12-2007	Expected
CXR-CS-20	Each central service shall report its internal status through the Data Acquisition Service [4.2].	A. Petrov 12-2007	Expected
CXR-CS-25	The central services shall be scalable downward to a portable minimalistic version capable to provide basic functionality for an autonomous control system.	A. Petrov 01-2008	Desired

4.1 Naming Service

Many classes of objects inside control systems are traditionally addressed by name. These can be front-ends, devices, events, datalogging channels, and error descriptors. Normally, the information about every entity is stored in a relational database, so it can be easily retrieved by applications. In practice, however, this can be affected by a number of issues:

- A limited amount of available database connections.
- The lack of direct connectivity due to security constraints.
- The need to know an exact structure of the database records for every class of objects, or an algorithm to extract the records from a legacy system.
- The data about a single class of objects can be stored in several separate databases.

A central Naming Service provides the clients with a set of characteristics for every named entry. For certain classes of objects, an implementation of a new data repository may not be necessary, because this information can already be retrieved from existing legacy systems. Regardless of how the data is actually stored, the Naming Service provides a uniform mechanism to lookup entries; browse, search, and modify the data.

No.	Requirement	Source	Priority
CXR-CS-30	The control system shall implement a central Naming Service providing arbitrary descriptive information about various classes of objects in the control system through a commonly accepted protocol.	A. Petrov 12-2007	Critical
CXR-CS-40	All entries shall be logically arranged in a tree.	A. Petrov 12-2007	Critical
CXR-CS-50	Each entry and intermediate node shall have a unique name.	A. Petrov 12-2007	Critical
CXR-CS-55	It shall be possible to create aliases (symbolic links) pointing to existing entries.	A. Petrov 01-2008	Expected
CXR-CS-60	Deleted entries shall be marked with a special attribute and preserved for future reference.	A. Petrov 01-2008	Critical
CXR-CS-70	For names of entries and intermediate nodes, the Naming Service shall mandate the character set, delimiter and escape characters, and a definition of uniqueness.	A. Petrov 12-2007	Critical
CXR-CS-80	At a minimum, each entry shall consist of a set of named characteristics.	A. Petrov 12-2007	Expected
CXR-CS-90	For characteristic names, the Naming Service shall mandate the character set.	A. Petrov 12-2007	Critical
CXR-CS-100	The characteristic values shall be type-safe.	A. Petrov 12-2007	Desired
CXR-CS-110	It shall be possible to specify a set of required and optional characteristics for different classes of objects.	A. Petrov 12-2007	Desired
CXR-CS-120	The Naming Service shall support, but not mandate, client authentication.	A. Petrov 12-2007	Critical

CXR-CS-130	It shall be possible to specify read and write permissions for a principal on a class of objects, and default permissions on all classes that are not the subject of explicit access control.	A. Petrov 12-2007	Expected
CXR-CS-140	Existing entries shall not be altered or reused in a way that invalidates the data stored in dataloggers and other long-term archives.	T. Bolshakov W. Marsh 01-2008	Expected
CXR-CS-150	The Naming Service shall mandate an entry serialization protocol.	A. Petrov 12-2007	Critical
CXR-CS-160	A client shall be able to retrieve any entry by name.	A. Petrov 12-2007	Critical
CXR-CS-170	A client shall be able to retrieve the set of children of any intermediate node.	A. Petrov 12-2007	Critical
CXR-CS-180	A client shall be able to search entries in a subtree by name, by presence of characteristics, and by characteristic values.	A. Petrov 12-2007	Critical
CXR-CS-190	A client shall be able to create new entries, modify characteristics of existing entries, and delete existing entries.	A. Petrov 12-2007	Critical

4.2 Data Acquisition Service

The purpose of a central Data Acquisition Service (DAQ) is to mediate in the real-time communication of data between the front-ends and the clients.

The front-ends provide data through individually addressable *devices*. Devices can be combined in groups to represent a machine, a subsystem, or a front-end. The *events* are generated by the control's timing infrastructure (XCLK, [3.1.1]), device state transitions, wall clock, and software. As the requirements do not prescribe any concrete model for devices and events, these terms here are provisional.

It is expected that the Data Acquisition Service may use different protocols to communicate with front-ends and with high-level applications, because the requirements (speed, security) on these levels may vary. However, it is desirable to use same data formats on both sides, so that the data samples [4.2.1] could come through the central layer transparently in either direction.

No.	Requirement	Source	Priority
CXR-CS-200	The control system shall implement a central Data Acquisition Service (DAQ).	A. Petrov 12-2007	Critical
CXR-CS-210	The DAQ shall provide data through a single Data Acquisition API (DAQ API). The DAQ API shall specify a device model, an event model, and data	A. Petrov 12-2007	Critical

	abstractions.		
CXR-CS-220	Each device in the control system shall be uniquely identified by its name.*	A. Petrov 12-2007	Critical
CXR-CS-230	The definition of devices shall be provided through the Naming Service.	A. Petrov 01-2008	Expected
CXR-CS-240	It shall be possible to express each event as a text string containing the event type, a unique event identifier within the type, as a set of parameters.	A. Petrov 12-2007	Critical
CXR-CS-250	The definition of XCLK events and software events shall be provided through the Naming Service.	A. Petrov 01-2008	Expected
CXR-CS-260	A client shall be able to acquire data from a device in one blocking operation.	A. Petrov 12-2007	Critical
CXR-CS-270	A client shall be able to acquire data from a device on an event in one blocking operation. If the event did not occur in a certain time frame, the call shall abort.	A. Petrov 12-2007	Expected
CXR-CS-280	A client shall be able to monitor a group of devices. A change of any device in the group shall cause a callback containing the new data.	A. Petrov 12-2007	Expected
CXR-CS-290	A client shall be able to monitor a group of devices on an event. The event shall cause a callback containing a snapshot of all the devices' data.	A. Petrov 12-2007	Critical
CXR-CS-300	A client shall be able to limit the maximum rate of the data returned through callbacks.	A. Petrov 01-2008	Expected
CXR-CS-310	A client shall be able to set data to a device in one blocking operation.	A. Petrov 12-2007	Critical
CXR-CS-320	A client shall be able to generate software events.	A. Petrov 01-2008	Critical
CXR-CS-330	A client shall be able to monitor a group of events. Each event occurrence shall cause a callback containing the event identifier and a timestamp.	A. Petrov 01-2008	Expected
CXR-CS-340	A client shall be able to set data synchronously to a group of devices using distributed transactions [CXR-LL-580].	A. Petrov 01-2008	Desired

4.2.1. Data Abstractions

In this document, an elementary quantum of data handled by the Data Acquisition Service is called *a sample*. Each sample represents an atomic reading from (or setting to) a front-end. At runtime,

* A device may also have one or multiple aliases, per CXR-CS-55.

samples exist as programmatic objects or structures specified in DAQ API. Whereas a particular format of these artifacts is not important for the requirements, it is essential that every sample can be rendered in three tangible forms:

- **Visual Representation,** a text string or an image suitable for human users.
- **Transport Format,** used to send the sample over network.
- **Persistent Format,** used to store the sample for a prolonged time.

The visualization function is specific to every class of objects. It needs to provide, perhaps, only a one-way transformation. The serialization mechanisms for transport and persistent formats shall be defined in the scope of the entire system, and have to work in both directions.

A sample consists of three parts: *a timestamp, a payload,* and *a status*. The Data Acquisition Service may parse and use the sample's timestamp and status. The payload is always application-specific and needs to be passed between front-ends and user applications unchanged.

The sample's payload is a collection of values and attributes. Each value represents the result of an actual measurement (for example, an ADC output) or a calculation, which can change over time. Multiple values in a single sample are allowed because some phenomena or processes have to be observed simultaneously at several points. The attributes provide descriptive information either about individual values (for example, a units text) or about the entire sample. Properties of the sample's data source, such as a device name, should not be included. The attributes are close to being constant, but they may occasionally change if the system is reconfigured. This fact must be considered in the design of the transport and persistent formats. For example, when samples are sent over network, some static information can be omitted to save bandwidth, providing that later on this data will be restored by other means. But when samples are archived, all their attributes shall be retained.

The DAQ API can provide multiple implementations of generic data types used as a payload, including a simple container holding one attributed value, an array of multiple attributed values, and more complex hierarchical formations. Also, the system can offer more specific structures for certain groups of applications, such as Wire Scanners. The clients must be able to properly interpret all applicable data types.

The sample's timestamp identifies a time when the measurement of the sample occurred. This can include the absolute calendar time, as well as time relative to certain system events (e.g., beam cycle).

The sample's status field [CXR-LL-415] indicates whether the reading operation was successful. It may also include additional information about conditions of the data acquisition (e.g., if the device has not responded properly) useful for a client .

For efficiency or logical harmony the control system may combine individual samples into *time sequential groups (TSG)*. Each TSG represents a temporal sequence of data as a whole. For example, a fast transient process recorded by a digital oscilloscope can be received on the client as a single TSG, rather than a long stream of individual readings. The TSG itself does not hold a timestamp and a status field.

No.	Requirement	Source	Priority
CXR-CS-400	Each distinct data sample handled by the Data	A. Petrov,	Critical

	Acquisition Service shall include a timestamp, a payload, and a status field.	R. Neswold 01-2008	
CXR-CS-410	The timestamp shall include absolute Coordinated Universal Time (UTC) of the sample.	A. Petrov 01-2008	Critical
CXR-CS-415	It shall be possible to include additional references in the timestamp, expressed as an amount of time passed since a certain instance of an event (using per-event counters [CXR-LL-90]).	A. Petrov 01-2008	Desired
CXR-CS-420	The DAQ shall be able to acquire samples in time sequential groups (TSG).	A. Petrov, B. Hendricks 01-2008	Critical
CXR-CS-430	A client shall be able to limit the maximum number of samples returned from a device in one call.	A. Petrov 01-2008	Expected
CXR-CS-440	The DAQ shall mandate serialization protocols for DAQ samples and TSG sent over network.	A. Petrov 12-2007	Critical
CXR-CS-450	The DAQ shall mandate serialization protocols for DAQ samples and TSG stored for prolonged time.	A. Petrov 12-2007	Critical
CXR-CS-460	The DAQ sample's payload shall be a collection of values and attributes. The values represent the results of actual measurements. The attributes contain descriptive information about individual values and the entire sample.	A. Petrov 12-2007	Expected
CXR-CS-470	The shall be a description of each device's data structures.	A. Petrov B. Hendricks 01-2008	Desired
CXR-CS-490	The DAQ shall not interpret or modify samples' payloads.*	A. Petrov 12-2007	Expected
CXR-CS-500	The DAQ API shall provide implementations of generic data types.	A. Petrov 12-2007	Critical
CXR-CS-510	The DAQ API shall provide implementations of data types, specific to common domains of applications.	A. Petrov 12-2007	Desired
CXR-CS-520	For each data type, the DAQ API shall support a mechanism of conversion to a human-readable format.	A. Petrov 12-2007	Critical
CXR-CS-530	The DAQ API shall explicitly define the use of special values, such as *Null*, *Infinity*, and *NaN* in samples, timestamps, payloads, and status fields.	A. Petrov 01-2008	Expected

* Note that the Data Logging Service may interpret and alter the payload when the data is returned to clients; see [4.4.1].

4.2.2.Acquisition of Alarms

Alarms are a special kind of data generated by the front-ends. Each alarm indicates an abnormal condition in the equipment, for example when device data is out of band. The Data Acquisition Service shall acquire alarms from the front-ends in the same way it acquires and processes any other data, with a few exceptions stated below.

No.	Requirement	Source	Priority
CXR-CS-540	The DAQ API shall provide a data abstraction for alarms. At a minimum, an alarm abstraction shall include an alarm state indicator and a bypass state indicator.	A. Petrov 12-2007	Critical
CXR-CS-550	Each alarm in the control system shall be uniquely identified by its name.	A. Petrov 12-2007	Critical
CXR-CS-560	The definition of alarms shall be provided through the Naming Service.	A. Petrov 01-2008	Expected
CXR-CS-570	A client shall be able to change the bypass state of an alarm device in one blocking operation [CXR-LL-730].	A. Petrov 12-2007	Critical
CXR-CS-580	Changes of the bypass state of an alarm device shall not trigger the monitors set by clients.	A. Petrov 12-2007	Critical
CXR-CS-590	If an alarm device is bypassed , changes of its alarm state shall not trigger the monitors set by clients.	A. Petrov 12-2007	Critical

4.2.3.Front-End Façade

It is hard to achieve a complete unification of low-level equipment in a real-world control system. Most likely, the entire collection of front-ends will not fully support one common set of functions and will not be able to speak exactly one communication protocol. The higher level components have to consider the equipment diversity and address it accordingly. As it would be too complex to design and too burdensome to maintain a large number of end-user applications able to talk individually to every piece of equipment, the front-end façade service takes over the responsibility to cast all front-end communications into one unified form.

No.	Requirement	Source	Priority
CXR-CS-600	The Data Acquisition Service shall support multiple types of front-ends and multiple protocols.	A. Petrov 12-2007	Critical
CXR-CS-610	The Data Acquisition Service shall support pluggable adapters to particular front-end types, linked dynamically at runtime.	A. Petrov 12-2007	Expected
CXR-CS-620	The semantic of each DAQ API call shall be constant, unrelated to the type of front-end targeted.	A. Petrov 12-2007	Critical
CXR-CS-630	The Data Acquisition Service shall emulate the	A. Petrov	Critical

	features of DAQ API missing from particular front-end implementations.	12-2007	

4.2.4.Data Redirection

Normally, a control system operates with devices connected to the real equipment front-ends. The function of data redirection replaces, transparently to clients, the entire set of real devices with a programmatically created one. There is a number of useful models that can be employed in data redirection, for example:

- **Setting Mirror** a snapshot of data represented in a memory model, that can be updated without addressing actual equipment.
- **Retrospection** a playback of data from an archive.
- **Physical Model** emulates behavior of the machine according to some theoretical principles.

No.	Requirement	Source	Priority
CXR-CS-650	The Data Acquisition Service shall support data redirection.	A. Petrov 12-2007	Critical
CXR-CS-660	The Data Acquisition Service shall support pluggable implementations of redirection models, linked dynamically at runtime.	A. Petrov 12-2007	Expected
CXR-CS-670	The semantic of each DAQ API call shall be constant, unrelated to the presence and the model of redirection.	A. Petrov 12-2007	Critical
CXR-CS-680	The common GUI shall include a visual indicator of the data redirection status.	A. Petrov 12-2007	Expected
CXR-CS-690	It shall be possible to operate data redirection from both the application and the central service.	A. Petrov 12-2007	Desired

4.2.5.Interoperability

Several issues arise from the fact that the entire Data Acquisition Service may consist of multiple programming instances distributed throughout the control system.

No.	Requirement	Source	Priority
CXR-CS-700	Remote links inside the DAQ shall be transparent. The control system shall attempt to recover lost connections and restore internal states of the peers.	A. Petrov 12-2007	Critical
CXR-CS-710	The DAQ shall automatically adjust to the actual number of running instances.	A. Petrov 12-2007	Critical
CXR-CS-720	It shall be possible to reconfigure the control system without a restart.	A. Petrov 12-2007	Expected
CXR-CS-730	Each DAQ instance shall provide an equal set of	A. Petrov	Expected

	functions for the clients, including access to the entire set of devices.	12-2007	
CXR-CS-740	A client shall not be required to choose which DAQ instance to connect.	A. Petrov 12-2007	Critical
CXR-CS-750	The control system shall remain operational if multiple nodes fail.	A. Petrov 12-2007	Expected

4.2.6. Data Consolidation

The volume of user requests to the low-level equipment in a control system is unpredictable. In different cycles some data becomes more important for the machine operation than others, thus attracting more traffic to the front-ends providing it. As some front-ends (or the network they are connected to) may not be capable of sustaining the high load, they can start returning data with delays or deny the service for some clients altogether. Data Consolidation shields the equipment from the excessive traffic by acquiring data from each front-end and caching it internally. All requests from the user applications and other central services are managed by DAQ data pools and do not address front-ends directly.

No.	Requirement	Source	Priority
CXR-CS-760	The Data Acquisition Service shall support data consolidation.	A. Petrov 12-2007	Critical
CXR-CS-770	The consolidation assignments among multiple DAQ nodes shall be defined dynamically.	A. Petrov 12-2007	Critical

4.2.7. Resource Locking

No.	Requirement	Source	Priority
CXR-CS-800	A client shall be able to set a lock on any device or a group of devices in the control system, in order to prevent settings from other clients.	A. Petrov 01-2008	Expected
CXR-CS-810	Each lock shall be associated with an authenticated client's principal.	A. Petrov 01-2008	Expected
CXR-CS-820	It shall be possible to specify an event (including a wall clock event), which will automatically remove the lock.	A. Petrov 01-2008	Desired
CXR-CS-830	A client shall be able to remove his own locks.	A. Petrov 01-2008	Expected
CXR-CS-840	A client with certain access privileges shall be able to remove any lock in the control system.	A. Petrov 01-2008	Expected
CXR-CS-850	A client shall be able to check whether a device or a group of devices is locked, and who owns the locks.	A. Petrov 01-2008	Expected

CXR-CS-860	If settings to a device are prohibited due to a lock, the client shall receive a clear error message referring to the lock owner.	A. Petrov 01-2008	Expected

4.2.8.Access Control and Audit

No.	Requirement	Source	Priority
CXR-CS-900	The Data Acquisition Service shall provide access control and support, but not mandate, client authentication.	A. Petrov 12-2007	Critical
CXR-CS-910	It shall be possible to specify read, set, and lock permissions for a principal on a single device, and on a group of devices. It shall be possible to specify default read, set, and lock permissions for a principal on all devices that are not the subject to explicit access control.	A. Petrov 12-2007	Expected
CXR-CS-915	It shall be possible to specify a permission for a principal to remove any lock in the control system.	A. Petrov 01-2008	Expected
CXR-CS-920	It shall be possible to specify permissions for a principal to generate an event, or a group of events. It shall be possible to specify default event permissions for a principal on all events that are not the subject to explicit access control.	A. Petrov 01-2008	Expected
CXR-CS-930	The Data Acquisition Service shall include a mechanism to disable settings from a client (*a setting lock*). It shall be possible to operate this mechanism from both an application and a central service.	A. Petrov 12-2007	Critical
CXR-CS-940	The common GUI shall include a visual indicator of the setting lock status.	A. Petrov 12-2007	Expected
CXR-CS-950	The Data Acquisition Service shall keep a log of connected clients.	A. Petrov 12-2007	Critical
CXR-CS-970	The Data Acquisition Service shall be able to identify potential misuse of the system by the clients, and keep a log of these events.	A. Petrov 12-2007	Expected

Note: Settings log requirement a given in CXR-LL-477.

4.3 Alarm Management Service

Alarm Management Service provides additional functions for aggregation and distribution of alarms in the control system.

No.	Requirement	Source	Priority
CXR-CS-999	The control system shall implement a central Alarm Management Service.	A. Petrov 01-2008	Critical
CXR-CS-1000	The Alarm Management Service shall implement a mechanism which combines multiple front-end alarms in a single aggregated alarm, using boolean expressions.	A. Petrov 01-2008	Critical
CXR-CS-1010	DAQ API shall operate with aggregated alarms in the same way it works with front-end alarms.	A. Petrov 01-2008	Expected
CXR-CS-1020	The Alarm Management Service shall provide a GUI for configuration of aggregated alarms.	A. Petrov 01-2008	Critical
CXR-CS-1030	The Alarm Management Service shall provide a mechanism for notifying users about alarms via email.	A. Petrov 01-2008	Expected
CXR-CS-1040	The Alarm Management Service shall support integration with notification mechanisms other than email.	A. Petrov 01-2008	Desired
CXR-CS-1050	The Alarm Management Service shall implement access control and mandate client authentication.	A. Petrov 01-2008	Critical
CXR-CS-1052	The Alarm Management Service shall keep a log of alarm.	A. Petrov 01-2008	Expected
CXR-CS-1054	For redundancy, the Alarm Management Service shall run in parallel on several nodes.	K. Krause 01-2008	Critical
CXR-CS-1056	The Alarm Management Service shall remain operation if multiple nodes fail.	A. Petrov 01-2008	Expected
CXR-CS-1058	The front-ends shall be able to push alarm to the Alarm Management Service.	J. Patrick 01-2008	Expected

4.4 Data Logging Service

No.	Requirement	Source	Priority
CXR-CS-1060	The control system shall implement a central Data Logging Service, which includes a number of multi-channel dataloggers.	A. Petrov 01-2008	Critical
CXR-CS-1070	Each datalogging channel shall store data from a single device, acquired on instances of a single event.	A. Petrov 01-2008	Critical
CXR-CS-1080	The configuration of dataloggers and channels shall be provided through the Naming Service.	A. Petrov 01-2008	Expected

CXR-CS-1090	Each datalogger shall be uniquely identified by its name.	A. Petrov 01-2008	Critical
CXR-CS-1100	Each channel within a datalogger shall be uniquely identified by the device and the event.	A. Petrov 01-2008	Critical
CXR-CS-1110	It shall be possible to assign an optional description and alias to a channel. Each alias shall be unique within the current datalogger.	A. Petrov 01-2008	Expected
CXR-CS-1130	The Data Logging Service shall support *direct* dataloggers, acquiring data from the Data Acquisition Service.	A. Petrov 01-2008	Critical
CXR-CS-1140	The Data Logging Service shall support *backup* dataloggers, acquiring data from other datalogging channels.	A. Petrov 01-2008	Critical
CXR-CS-1150	The Data Logging Service shall support *client* dataloggers, accepting data from arbitrary external processes.	A. Petrov 01-2008	Critical
CXR-CS-1160	The Data Logging Service shall mandate authorization of processes submitting data to the client dataloggers.	A. Petrov 01-2008	Expected
CXR-CS-1170	The Data Logging Service shall log device access errors in the same way it logs successful samples.	J. Patrick 01-2008	Expected
CXR-CS-1180	The Data Logging Service shall provide data through a single Data Logging API (DL API). DL API shall reuse the device model, the event model, and the data abstractions from DAQ API.	A. Petrov 01-2008	Critical
CXR-CS-1190	A client shall be able to acquire data from a datalogging channel in one blocking operation.	A. Petrov 01-2008	Critical
CXR-CS-1200	A client shall be able to acquire data from a datalogging channel through asynchronous callbacks.	T. Bolshakov 01-2008	Expected
CXR-CS-1210	The Data Logging Service shall output data in TSG*: ● If the channel acquires TSGs, it shall return the same TSGs to clients. ● If the channel acquires discrete samples, it shall pack them into TSGs.	A. Petrov 01-2008	Expected
CXR-CS-1220	A client shall be able to limit the maximum number of samples returned from a channel in one call.	A. Petrov 01-2008	Expected
CXR-CS-1230	A client shall be able to start/stop data acquisition in direct and backup dataloggers, and enable/disable	A. Petrov 01-2008	Critical

* Time sequential group, see [4.2.1].

	client dataloggers.		
CXR-CS-1240	For a given device, a client shall be able to obtain the list of associated direct and backup datalogging channels.	A. Petrov 01-2008	Critical
CXR-CS-1250	For a given datalogging channel, a client shall be able to obtain the associated device and event, and a timestamp of the oldest data sample.	A. Petrov 01-2008	Critical
CXR-CS-1260	The Data Logging Service shall implement access control and support, but not mandate, client authentication.	A. Petrov 01-2008	Critical
CXR-CS-1270	It shall be possible to specify reading and operational (start/stop) permissions on a datalogger, and default permissions an all dataloggers that are not the subject of explicit access control.	A. Petrov 01-2008	Expected

4.4.1. Transformations of Archived Data

No.	Requirement	Source	Priority
CXR-CS-1280	The data logging system shall not alter the data stored in the dataloggers.	A. Petrov 01-2008	Expected
CXR-CS-1290	The data logging system shall provide a mechanism to correct data returned from the dataloggers, in order to offset past measurement inaccuracies and replace obsolete data formats.	K. Cahill 01-2008	Critical
CXR-CS-1300	The data logging system shall be capable to store a historical list of correction functions for each device.	T. Bolshakov 01-2008	Critical
CXR-CS-1310	A client shall be able to disable the correction function.	T. Bolshakov 01-2008	Critical
CXR-CS-1320	A client shall be able to specify a filter to be applied to the data acquired from a dataloggers. (For example, if a datalogger stores arrays of data or some complex data structures, a client may want to extract only one element rather than the entire structure).	T. Bolshakov 01-2008	Desired

4.5 Hierarchical Logging Service

The Hierarchical Logging Service (also known as Sequenced Data Acquisition, SDA) is a multi-stage event-driven datalogging system, which records only the data that are essential on a particular stages of machine operation. A generalized model of hierarchical datalogging is shown on Fig. 1.

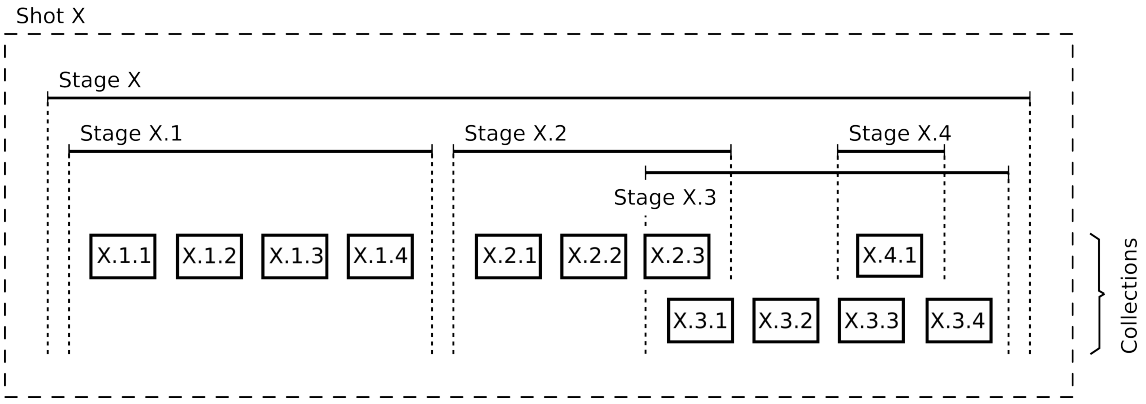

Fig. 1: Model of Hierarchical Datalogging

The hierarchical datalogging operates with *stages*. Each stage is associated with a number of events. The events specify conditions upon which a particular stage starts and finishes. The stages are arranged in a trees. When a stage starts, it enables all stages on the next level, so they can start too when appropriate events will occur. When a stage finishes, it disables all the underlying stages, possibly terminating those that are still running. Stages on one level can run in parallel.

The stages on the bottom of the hierarchy are called *collections*. Each collection is an equivalent of a multi-channel datalogger [4.4]. Yet, the collections usually store much less data and may have different implementations.

The definition of every single top-level stage, along with all stages and collections lying underneath it, constitute a *shot*. When the top-level stage of a shot is triggered, the system starts moving through a sequence of events, and can ultimately reach collections that will record the necessary data. All stage instances (i.e., stages that have already happened in the past) and collection instances are individually addressable from the top to the bottom of the hierarchies.

This requirements do not constrain an implementation of hierarchical dataloggers by prescribing one invariable model. It is assumed that the control system may have more than one type of hierarchical logging service, which utilize different mechanisms of storing and accessing the data, and different state transition rules. In particular, there may be a separate way of dealing with small frequent shots (also known as *pulses*).

No.	Requirement	Source	Priority
CXR-CS-1330	The system shall implement a central hierarchical logging service.	A. Petrov 01-2008	Critical
CXR-CS-1340	Each shot shall consist of a tree, which includes stages and collections.	T. Bolshakov 01-2008	Expected
CXR-CS-1350	The number of level for each shot shall be constant.	T. Bolshakov 01-2008	Expected
CXR-CS-1360	The stages and collections shall be started and stopped on events. If a stage is running, it enables all stages and collections underneath.	T. Bolshakov 01-2008	Critical
CXR-CS-1370	Each collection shall be a functional equivalent of a	A. Petrov	Expected

	multi-channel datalogger [4.4].	01-2008	
CXR-CS-1380	Each stage and collection shall have an index, unique on the current level within the shot.	T. Bolshakov 01-2008	Critical
CXR-CS-1390	It shall be possible to assign an optional description and alias to each stage or collection. The alias shall be unique on the current level within the shot.	T. Bolshakov 01-2008	Expected
CXR-CS-1400	The definition of shots and a list of available collections shall be provided through the Naming Service.	A. Petrov 01-2008	Expected
CXR-CS-1410	SDA shall implement access control and support, but not mandate, client authentication.	A. Petrov 01-2008	Critical
CXR-CS-1420	It shall be possible to specify permissions to reconfigure shots, permissions to read data from collections that belong to a particular shot, and default reading permissions on all shots that are not the subject of explicit access control.	A. Petrov 01-2008	Expected

4.6 Postmortem Logging Service

In this section, the term *postmortem logging* refers to the acquisition of data immediately preceding a failure of the system in a large scale (e.g., accelerator quench). It does not cover various specialized postmortem tools, such as operating systems' core dumps or front-end debugging facilities.

No.	Requirement	Source	Priority
CXR-CS-1430	The control system shall implement a central Postmortem Logging Service, which orchestrate the collection and distribution of postmortem data.	A. Petrov 01-2008	Critical
CXR-CS-1435	The Postmortem Logging Service shall use a snapshot on event function [CXR-LL-530] on front-ends. (A snapshots may include the data before the event, or the data before and after the event).	A. Petrov C. Briegel 01-2008	Expected
CXR-CS-1440	The Postmortem Logging Service shall emulate the snapshot on event function, if it is missing in a front-end.	A. Petrov 01-2008	Expected
CXR-CS-1445	The Postmortem Logging Service shall be responsible for the snapshot configuration in front-ends.	A. Petrov 01-2008	Expected
CXR-CS-1450	Upon a critical event, the Postmortem Logging Service shall collect data from front-ends and store it in a central datalogger.	K. Cahill 01-2008	Critical

CXR-CS-1460	Upon collection of postmortem data, the Postmortem Logging Service shall trigger an event to notify the clients.	K. Cahill 01-2008	Critical
CXR-CS-1470	Each postmortem logging channel shall be uniquely identified by the associated device.	A. Petrov 01-2008	Critical
CXR-CS-1480	The configuration of each channel shall be provided through the Naming Service.	A. Petrov 01-2008	Expected
CXR-CS-1490	A client shall be able to read postmortem data via DL API [4.4].	A. Petrov 01-2008	Expected

4.7 Save And Restore Service

No.	Requirement	Source	Priority
CXR-CS-1500	The control system shall implement a central Save And Restore Service.	A. Petrov 01-2008	Critical
CXR-CS-1510	Each Save And Restore archive shall store snapshots of data samples acquired synchronously from a group of devices.	A. Petrov 01-2008	Critical
CXR-CS-1520	The configuration of archives and the list of available snapshots shall be provided through the Naming Service.	A. Petrov 01-2008	Expected
CXR-CS-1530	Each archive shall be uniquely identified by its name.	A. Petrov 01-2008	Critical
CXR-CS-1540	Each snapshot within an archive shall be uniquely identified by the timestamp.	A. Petrov 01-2008	Critical
CXR-CS-1550	It shall be possible to assign an optional description and alias to a snapshot. The alias shall be unique within the current archive.	A. Petrov 01-2008	Expected
CXR-CS-1560	A client shall be able to obtain the list of available snapshots in an archive.	A. Petrov 01-2008	Critical
CXR-CS-1570	A client shall be able to obtain an individual snapshot from an archive.	A. Petrov 01-2008	Critical
CXR-CS-1580	A client shall be able to save a new snapshot into an archive.	A. Petrov 01-2008	Critical
CXR-CS-1590	A client shall be able to restore a previous state of the system by using an archived snapshot.	A. Petrov 01-2008	Critical
CXR-CS-1600	A client shall be able to modify data in the existing snapshots.	K. Cahill 01-2008	Expected

CXR-CS-1610	The modified snapshots shall be clearly marked with a special attribute.	K. Cahill 01-2008	Critical
CXR-CS-1620	The Save And Restore service shall be able to take snapshots automatically on events.	A. Petrov 01-2008	Critical
CXR-CS-1630	The Save And Restore Service shall implement access control and support, but not mandate, client authentication.	A. Petrov 01-2008	Critical
CXR-CS-1640	It shall be possible to specify reading, saving (make a new snapshot), restoring (restore data from a snapshot) and writing (modify a snapshot) permissions on an archive, and default permissions an all archives that are not the subject of explicit access control.	A. Petrov 01-2008	Expected