Document Title:	**Detailed Specification of the FAIR Accelerator Control System Component "Archiving System"**
Description:	This document is the Detailed Specification of the accelerator control system component 'Archiving System'. Its task is to collect and store all pertinent accelerator data centrally to facilitate the analysis and tracking of the accelerator performance as well as its proper function.
Division/Organization:	CSCO
Field of application:	FAIR Project, existing GSI accelerator facility
Version	V 4.5

Prepared by:	**Checked by:**	**Approved by:**
V. Rapp L. Hechler R. Steinhagen	**FAIR-C2WG-ALL** A. Reiter (BI) M. Schwickert (BI) J. Fitzek (CO) S. Reimann (OP) P. Schütt (OP) C. Omet (SIS-100 MP) D. Ondreka (System Planning) I. Lehmann (Machine-Exp.) D. Severin (Machine-Exp.) **MPLs & MCs***	R. Bär (Controls) R. Steinhagen (FAIR Comm. & Control)

*List of Machine-Project-Leaders (new FAIR acc.) & Machine Coordinators (existing facility):

F. Hagenbuck (HEBT), R. Bär (Controls), M. Winkler (Super-FRS), O. Dolinskyy (CR), P. Spiller (SIS-100), K. Knie (p-Linac & p-bar Separator), R. Steinhagen (FAIR Comm. & Control), R. Hollinger (Ion Sources), P. Gerhard (UNILAC), J. Stadlmann (SIS-18), M. Steck (ESR), F. Herfurth (CRYRING / HITRAP).

Document History:

Version	Date	Description	Author	Review / Approval
0.1	06. Oct 2011	Draft	L. Hechler	
1.0	07. Oct. 2011	Draft version	CCT	CCT
1.1	17. Oct. 2011	New structure	L. Hechler	
2.0	26. Oct. 2011	Final version	CCT	CCT
2.1	17. Nov. 2011	Renaming of referenced guidelines	CCT	
3.0	03. Aug. 2012	Incorporated FAIR review comments	CCT	EDMS
4.1	31. Mar. 2016	Reworking and new Draft version	V. Rapp	
4.2	20. May 2016	Incorporated comments and feedback from the draft-review process	V. Rapp	
4.3	6. June 2016	- Feedback from the FC2WG - Added aspects related to Apache Cassandra and Kafka (Messaging) - incorporated feedback from - external reviewer (Mr. Marsching) - J. Fitzek (LSA & GUI related)	V. Rapp	
4.4	29. June 2016	- FC2WG Approval - Finalizing Document	V. Rapp	FC2WG
4.5	18. July 2016	- changed to newer QA template - reformatting - preparation in view of engineering check - then approval via EDMS	R. Steinhagen R. Bär	EDMS

Table of Contents

1 Purpose and Classification of the Document

This document specifies the functional requirements and the applicable technology choices for the Accelerator Control System component "Archiving System" for FAIR (PSP code 2.14.10.2.9).

For this component, this is the most detailed document in the hierarchy of Control system specifications.

Whenever regulations and requirements are specified in the General Specifications, Technical Guidelines or Common Specifications of the Control System, they are only referenced in this document. The related documents are listed in the Appendix.

No legal or contractual conditions are treated in this document. All related information is given in the General Specifications for FAIR.

1.1 Responsibilities

The responsibilities with respect to changes and modifications of the present document are entirely in the hands of the Control System Project Department of the GSI Helmholtz Centre for Heavy Ion Research GmbH (GSI) Darmstadt.

For initial information please contact the administration of the Controls Department.

Further information on the organisation chart, names of responsible persons and task leaders, as well as the agreed document release and approval procedure is summarised in the organizational note 'Controls Project for FAIR'.

1.2 Classifications of Requirements

The following definitions of requirement classifications are being used throughout the document:

- "**Must**" or "**shall**" or "**is required to**" is used to indicate mandatory requirements, strictly to be followed in order to conform to the standard and from which no deviation is permitted.
- "**Must not**" or "**shall not**" mean that the definition is an absolute prohibition of the specification.
- "**Should**" or "**is recommended**" is used to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others or that a certain course of action is preferred but not required.
- "**Should not**" or "**is not recommended**" mean that there may exist valid reasons in particular circumstances when the particular behaviour is acceptable or even useful, but the full implications should be understood and the case carefully analysed before implementing any behaviour described with this label.
- "**May**", which is equivalent to "**is permitted**", is used to indicate a course of action permissible within the limits of the standard.

2 Scope of the Technical System

2.1 System Overview

The purpose of the Archiving System is to store historical data gained and generated by the individual accelerator components as well as control system in a permanent location. It allows storing data on a configurable base of resolution in time, triggered by timing events or on-change. Data may be either gathered from devices, higher level data like computed physics properties or generated abstract data (see e.g. [11]).

The Archiving System includes functionality to query, filter, correlate and display historical data. For data identification and ordering purposes the system should be able to query, receive and store data from other control sub-systems such as the accelerator's settings management system (LSA), the Beam Transmission Monitor system or the Post-Mortem system.

The Archiving System includes services that allow data reduction over time or aggregation of historical data on a configurable base.

The Archiving System provides control room applications to display, configure, and manage archived data.

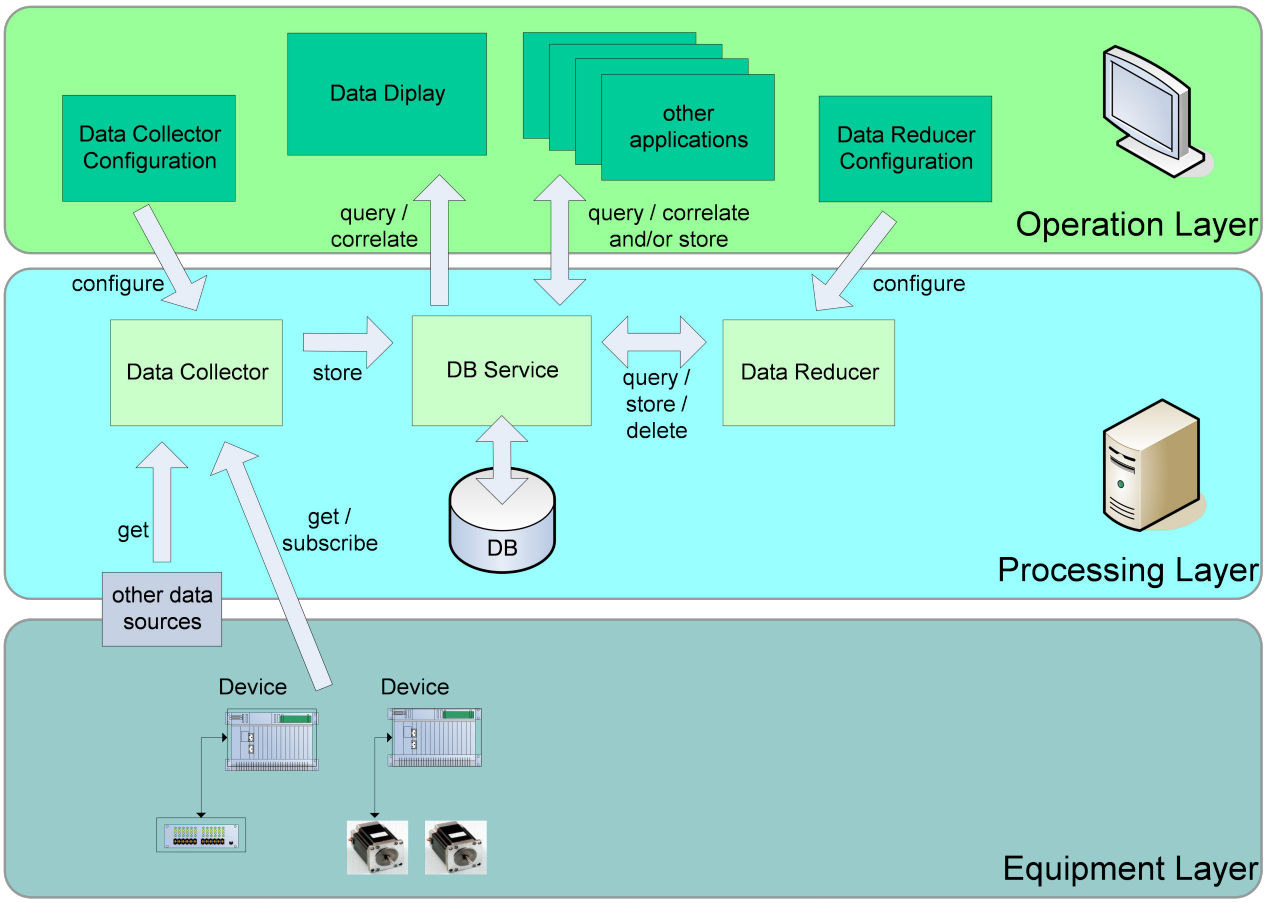

Figure 1: System Overview

2.2 Abbreviations, Terms and Definitions

A list of common controls related abbreviations used at GSI and FAIR can be found at [1]. Abbreviations specifically used in this document are:

Abbreviation	Definition
Accelerator Mode	Enum-type controls variable describing deliberate user-driven state used to pre condition the control system behaviour, covering rule sets outside of beam operation, defined per accelerator/transfer-line segment. Examples: SHUT-DOWN, ACCESS, MACHINE-CHECKOUT, PHYSICS, ...
Beam Mode	Enum-type controls variable describing deliberate user-driven state used to pre condition the control system behaviour, covering rule sets during beam operation, defined per accelerator/transfer-line segment and Beam-Production-Chain. Examples: NO-BEAM, PILOT, ADJUSTING, STABLE-BEAMS, ...
BPC	Beam Production Chain, an organisational control system structure to manage parallel operation and beam transfer through the FAIR accelerator facility. It describes a 'beam' through the facility including the sequence and parameters of beam lines and accelerators, starting from the ion-source up to an experimental cave (e.g. APPA, CBM, Super-FRS, ...). The BPC structure includes the definition of target beam parameters (set values) like, for example, isotope type, energy per nucleon, charge per nucleon, peak intensity, etc.
BTM	Beam Transmission Monitoring System
CMW	Common Middle-Ware (communication protocol between FECs & clients)
DevAcc	Device Access (GSI's old legacy communication middleware and FEC infrastructure)
FEC	Front-End Controller
FESA	Front-End Software Architecture
JAPC	Java API for Parameter Control (communication protocol on to of CMW)
PM	Post-Mortem System
UNICOS	Unified Industrial Control System (vacuum controls, cryogenic controls, etc.)

2.3 Limits of the System and Environment

2.3.1 Limits

The Archiving System does not cover the following:

- accelerator settings generation, administration and storage, covered by the existing LSA settings management system [3]
- diagnostic logging data storage, covered by [16]

However, due of similarity of data storage concepts the design of the Archiving System shall aim to provide storage functionality for following systems:

- Beam Transmission Monitoring system (BTM) [5]
- Post-Mortem system (PM) [8]

2.3.2 Interfaces

The Archiving System shall have an interface to control system devices [6,7], using JAPC (Java).

In order to correlate and display archived data, control room applications linked to the Archiving System must be able to query data from the accelerator's settings management system [2], the Beam Transmission Monitor system [4], and the Post-Mortem system [5], as well as the UNICOS systems data storage system for short-term data.

Additional interface to the FESA Database must be implemented. This should be used to track renaming of devices, in order that the extraction service can retrieve information for given devices, even if they were renamed or their interfaces changed in the past.

Additionally, the Archiving System should implement a database-to-database transfer from the WinCC OA (UNICOS) System for long-term data storage.

2.3.3 System Environment

The Archiving System runs on dedicated Linux and database servers. The detailed system environment and configuration is specified by the Control System Department by the time the implementation shall begin (defined as a project milestone).

The IT hardware (servers) and software product licenses for the production system at FAIR will be provided by the Company.

2.4 Basis of Concept

2.4.1 Functional Requirements

Number	Description of Requirement

AR_010	The Archiving System must archive data on a long-term basis for at least 10 years or over the whole life-cycle of the full accelerator facility.
AR_020	The data and their sources to be archived must be freely configurable without the necessity to start/stop the Archiving System.
AR_030	Archiving configurations must be storable for re-use later.
AR_040	Different archiving configurations must be supported: • default configuration that is valid at the start of the Archiving System • adjustable predefined configurations that are used regularly and are available at the “push of a button“ • some critical archiving system parameters shall be static and protected from dynamic configuration profile changes • custom configurations created by users
AR_050	The Archiving System must offer supporting functions to query and correlate archived data. Those queries include: • select data for a particular property for a given time period • select time periods where a given parameter equalled, dropped below or exceeded a particular value • select all time periods with specific properties (e.g. with post-mortem events, particular ion-species, actual intensity above given value etc.)
AR_060	The Archiving System must offer a service to reduce or delete archived data automatically or manually.
AR_070	Automatic reduction or deletion of archived data must be freely configurable.
AR_080	Configurations for reduction and deletion must be storable for later re-use.
AR_090	The Archiving System must offer an API • to write data into the archive database, • to read data from the archive database, and • to help correlate data read from the archive database. The API must provide asynchronous calls and should be implemented in Java. An additional C++ implementation of the client library is desirable. A RESTful interface must be considered to use in the server side extraction service.
AR_100	The Archiving System must provide a GUI application to configure the system, to correlate and display archived data.
AR_110	The Archiving System must be able to export (correlated) archived data in a generic format (e.g. comma separated value files (csv), ROOT) to support further post-processing using third-party applications and scripts. The primary user-level interface language should be implemented in Java with a function-equivalent secondary interface using C++ (further

	permitting integration of other non-Java, e.g. scripting languages, etc.)
AR_120	The Archiving System shall provide facilities to store meta data alongside the regular data that can be used to index, preselect and filter the data to be queried and extracted from the database. Some of the meta data include: targeted beam parameter, machine parameter, Accelerator- & Beam Modes, whether PM trigger has been issued, operational ranges (OP day, week, time between technical stop), and important operational markers (start/end operation, start/end technical stop, …),
AR_130	The architecture of the Archiving System shall allow an easy extension to propagate the collected data to 3rd party systems. Such extension could be a data proxy component, which itself provides other systems with live data collected from the devices, and subsequently reducing the load on the front-end controllers.
AR_140	Following the requirement AR_130 the proxy extension must be able to provide data to the users outside of the accelerator network.
AR_150	Load management and monitoring of the Archiving System infrastructure must be supported.
AR_160	User access management (disconnecting/limiting of “piggy” clients, bandwidth shaping/scheduling etc.) must be included.

Table 1: List of Functional Requirements

2.4.2 Non-functional Requirements

The archiving system must be designed and implemented to be scalable in terms of performance of data acquisition. The expected maximum data traffic volume at the moment of writing of this documentation is described in following numbers. These values however must be seen as rough estimates for minimum requirements of the system, which should be able to scale during the operation period.

Parameter	**Value**
Amount of monitorable devices	2032
Total amount of monitorable variables	~50000
Incoming data load	~30 MB/s
Expected outgoing user-level data load	up to 300 MB/s
Required data buffer (short time storage)	~5 – 10 TB/week
Required data storage (long term storage)*	5 – 10 TB/year

**Estimated data reduction of 25-50x*

Table 2: Non-functional Requirements

Note that the provided estimations do not represent the actual storage space required on disk, as this depends strongly on diverse technical parameters used to store the data later

such as: data format, database technology, replication factor etc. Hence the final estimation can only be provided after a full definition of all these parameters.

2.4.3 General Constraints

Any language specific localisation of accelerator components must comply with the accelerator's control system guidelines.

Any user or rights management must be designed and developed in accordance with the central rights management within the control system.

All GUIs must comply with the GUI Guidelines [9].

2.4.4 Architectural Principles

The Software Architecture Guideline for the Control System [10] fully applies.

3 Technical Specifications

The Archiving System consists of various components. The main parts of the system are presented in Figure 12.

There are two possible approaches to how the data is collected and aggregated by the system. First is the active subscription to different measurement and status parameters of configured or known devices. Those devices would be typically implemented using FESA and hence the subscription can be established using JAPC. Other data to be stored are from UNICOS devices whose interface will be defined a later stage. Another form of data aggregation is an active data supply. Therefore the Archiving System provides an API, which can be used by various clients to archive and store their own data in the system ('push' interface).

The automatically collected data is verified by the Data Inspector for its validity (e.g. correct input range, input data rate, data formats, etc. but not necessarily data-based content filtering) and afterwards written to the database using the DB Writer.

The Database can be defined into short- and long-term storage. While short-term data must be available in its full granularity, however only for a given time period, long-term data should be available for the accelerators lifetime, but may be compressed.

The data reduction service reduces historical data and writes them on to the long-term storage.

Data aggregation, checking and reduction can be configured using an administration GUI.

The Data Extractor is used to access the stored data. It provides an API to 3rd party applications as well as an own user-level GUI, used to present the data in various formats for future analysis. Therefore the Data Extractor finds correlated records inside the long- and short-term DB and delivers them to the user.

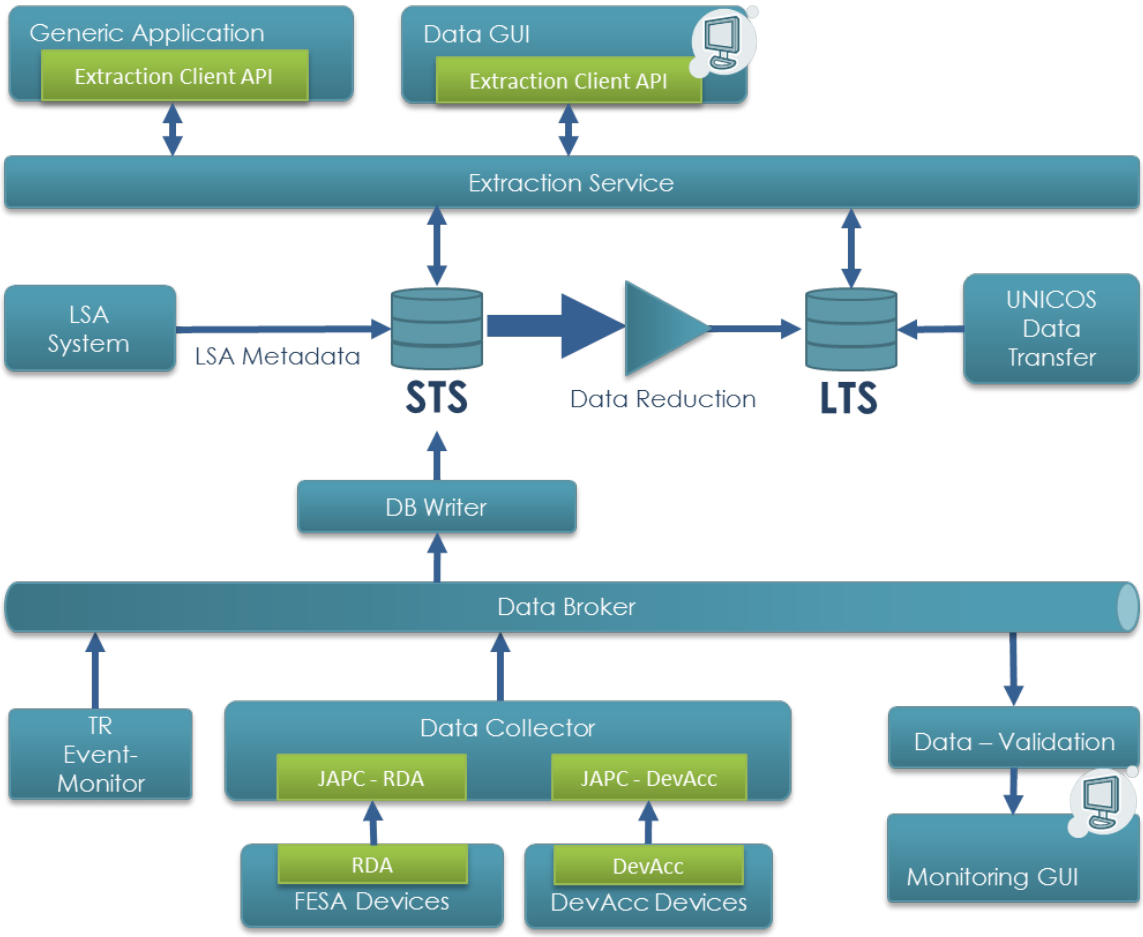

Figure 2: Components Overview

3.1 Storage and Access

A database service should be used to store the incoming data.

There are two favourite databases at the start of the project: *Apache Cassandra* [13] and *Elasticsearch* [17]. Both needs to be evaluated during first releases and the prototyping phase in terms of their write and read performance as well as other functionalities. A final decision about the used technology may only be done after the evaluation process. Therefore the architecture must be flexible enough to overcome the possible change of the used DB.

The final decision about the database in the final release shall be made as soon as possible after the evaluation and the first experience with Apache Cassandra.

The Oracle Database system for storage may be considered as a main fallback option in case the other NoSQL-based database solutions prove themselves as unsuitable for GSI/FAIR.

3.1.1 Data Reduction

The Archiving System must be able to automatically reduce historical data. This reduction process must be configurable, see chapter 3.1.2.

For special data reduction requirements it must be possible to easily add new algorithms via plug-in-type mechanisms. Therefore an interface for those plug-ins must be implemented. The system must be able to scan, find and integrate new plug-ins at run-time.

The following reduction options must be available in the archiving system from the start:

- No Reduction: AS does not apply a data reduction
- Reduced Sampling Rate: AS stores data with a reduced sampling rate i.e. store each n^{th} value (for example for a data reduction from 1 kHz to 100 Hz)
- On Change: AS stores only value changes, which exceed a certain threshold. The threshold should be adjustable for single parameters, as well as a whole system; a periodic snap-shot of the variable should be stored regardless in case the data has not changed within a pre-defined time period
- Timed Aggregation: Aggregate and store values for multiple Beam Production Chains (BPCs) for a single parameter:
    - Minimum value (for each time-stamp within one BPC)
    - Maximum value
    - Standard deviation (σ, stdev)
    - Median value
    - Mean value
    - Transient values – actual values outside of a 'n·σ' band
    - One random event during this period in order to reduce/check the filter bias

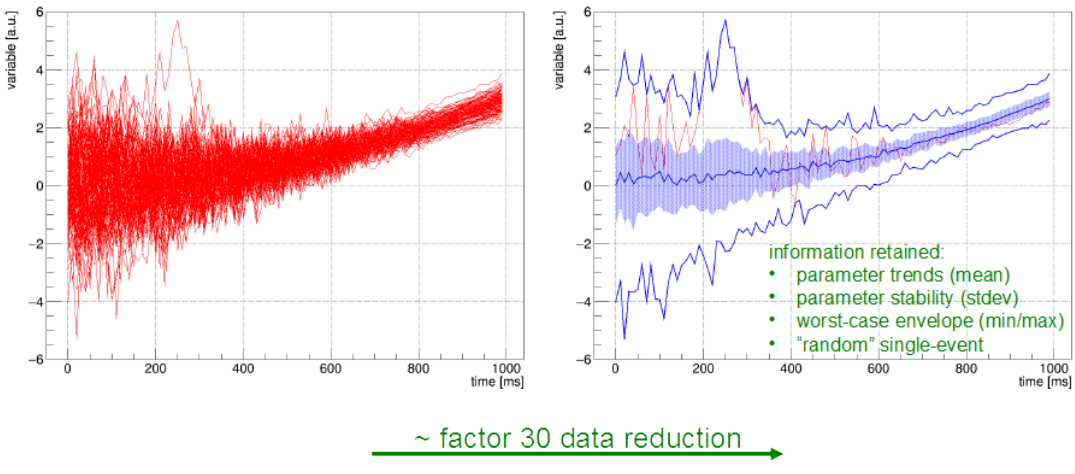

Figure 3: Timed Aggregation

It must be possible to configure the exceptions of data reduction i.e. data which should be only compressed using lossless compression (e.g. not compressed at all). Examples of such values include beam profiles at injection and extraction or postmortem data.

A compression rate of factor 20-50x must be targeted. However, the actual required rate can only be determined during the prototype test and evaluation phase, as it strongly depends on database technology, its storage capability and the data model behind.

The implementation of the reduction algorithm must consider the specific issues of the underlying database. Therefore this process must be additionally evaluated and taken into account during the data model design. For example, a Cassandra Database has known performance drawbacks on particular data removal.

Due to performance reasons the reduction process may be realised as a separate parallel process. In this case it collects the data similar to the data writer (3.2.4) directly from the underlying broker, aggregates values as described and writes them directly into the long term database storage. At the same time the short-term storage entities should be provided with an obsolescence ('time-to-live') counter, which regulates how long the data is kept.

Additionally the reduction process may be implemented as cascaded multi-step process. In this case the aggregation periods shall be progressive, with the data age. E.g. aggregate over 1 minute for data older than 1 month in first step and aggregate the data over 10 minutes for data older than 6 month in the second step. Such a cascaded data reduction is also beneficial for the user-level data extraction process, where users typically require first coarse data to pre-select a given time period before requiring more and more specific and high-time-resolution data sets.

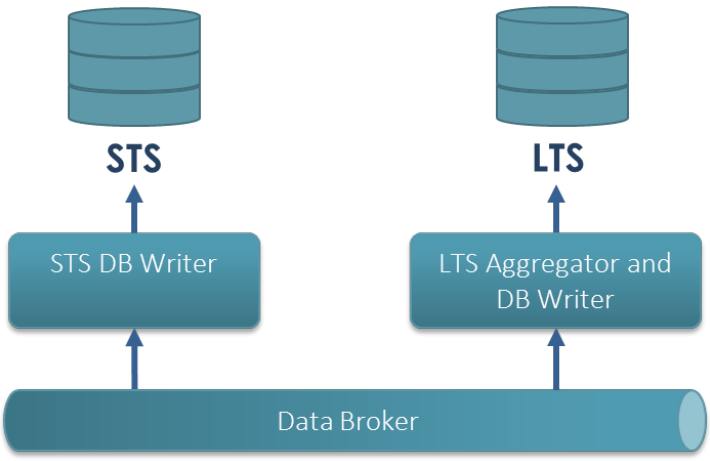

Figure 4: Alternative approach for Data Reduction

3.1.2 Data Reduction Configuration

Automatic reduction of historical data must be configurable. The following points must be adjustable:

- when or how often data should be reduced or deleted

- which algorithm should be used to reduce data
- which data should be reduced or deleted
- reduction rate

Parameters should be adjustable depending on the 'accelerator-' and 'beam mode' of the accelerator or transfer-line segment for device associated with them.

3.1.3 Data Extractor

The Data Extractor component is responsible for querying archived data. It consists of two parts – the server (database) and the client library. The server encapsulates the database requests, such as that the user is not (required to be) aware of how or where the data is stored. Another part should be implemented in form of client library, which hides the communication details between the server and the client components.

The client library must be implemented in Java. A C++ API with the same functionality should be implemented. It must provide functionality to:

- query data from the archive database
- find correlated entries in the queried data

The server should offer a clear and well-documented interface to query the data. A generic service (e.g. RESTful) should be considered as the main option. The client component may utilise this service, but can also be based on other communication patterns. However, other system components should be designed with a proper abstraction level, allowing an easy exchange of the underlying messaging layer (e.g. in case, the RESTful interface does not fulfil the performance requirements).

Since the server should directly communicate with the database, its implementation highly depends on the data storage technology used. Depending on the type of database there are multiple tools and languages available that should be evaluated before starting the actual implementation. Especially for Apache Cassandra the usage of Spark [15] must be evaluated.

The extractor's server component must be easily scalable with regard to the amount of requests per second. It must be guaranteed that when extracting data the writing performance of the system remains sufficient to handle the whole write load. The writing has a higher priority over data extraction.

3.1.3.1 Extraction Use Cases

Some typical planned use cases of the Archiving System to illustrate the usage of the data extractor:

1. A user selects a particular value or a set of values to be extracted, viewed and analysed, defined for a given time period (given t_{start} and t_{stop}, last day, last month etc.)

2. A user selects a time period but specifies additional selection parameters. Such parameters can be the following:
    - Show values only for specific context
    - Show only beam production chains with beam
    - Retrieve data only between specific events (e.g. injection -> extraction)
    - Select values for a given beam types (e.g. specific element beams)
3. As described before, but the system allows a selection of a particular BPC with specific parameters. These parameters include:
    - Beam parameter
        - Ion-species
        - Beam target (experiment)
        - Actual intensity
        - Actual beam transmission
        - Targeted intensity range
    - Accelerator- and Beam-Modes (enum-style variables, see section 2.2)
    - Machine parameter
        - Ramp-rate
        - Min/max rigidity
        - Injection/extraction energy
        - Slow/fast extraction
        - Cycle/store length
    - Beam Production Chains with(-out) post-mortem events
    - Operational ranges
        - OP year
        - Between technical stops
        - OP week
        - OP day

A combination or logical cascading of described filter criteria should be possible as well.

3.1.4 Database Data Structures

For data querying and correlation, stored data set must at least contain the following information:

- time stamp
- accelerator context (selector)

- unique signal source identification, like:
  - nomenclature, property name, field name
  - host name, service name, class name
- SI unit

Data sets must at least support all JAPC (FESA) data types.

The main structuring key for the data inside of the Archiving database must be the device-property value. Each collected entity must also contain a time stamp defining when it was collected as well as received selector information, which (among the time stamp) identifies to which beam production chain the information belongs. Additional identification or clustering keys may be added, due to restrictions or performance considerations of the underlying database.

Among the normal collected values, the database must keep track of some dedicated parameters, whose actual values can be used for selections. Such selection queries may look as follows: select all intervals or beam production chains where a value of selected dedicated parameters equals (greater, less than) a given value. Such dedicated values are not subjects of data reduction. The transfer of the meta-data from the LSA- as well as the timing system and some other parameters should be realised using this approach.

Device and parameter names are not static in the GSI environment, and may change over time. In this case, the extractor component needs to know how to access a particular value in the past. The history of the renaming of devices or change of interfaces is kept in the FESA database. The FESA database also keeps track of all running devices and their properties. Unfortunately, this information is only available for the FESA devices, while older DeviceAccess devices are not using a particular development database. Hence the renaming information must be kept inside the archiving database and provided manually. Therefore a small GUI should be developed to write the rename information for DeviceAccess devices.

The Archiving System must support storage of large binary objects, such as pictures or even short video sequences. Possible technical solutions (e.g. storing only Links inside of DB or whole objects etc.) in combination with the database must be evaluated.

The final data model and its storage concept (e.g. storing data as JSON objects) shall be considered as part of the implementation work.

3.1.5 UNICOS DB Data transfer

Besides the device data published via JAPC, the archiving system should also archive the information from WinCC OA (UNICOS) industrial control SCADA systems. These systems will use their own database (Oracle or Raima). Since these databases will not be used for long-term usage, a transfer to the archiving system is required.

3.1.6 Additional Data (Meta-information)

In order to select the stored data from devices according to the user's needs the archiving system shall store some historical information from other systems from the GSI controls environment.

3.1.6.1 Timing Information

Timing information is required to assign the data to particular timing periods with specific properties (e.g. beam-on, beam-off). This information contains timing group-ID and event names with corresponding time stamps.

A timing receiver component should therefore listen to timing events and propagate them to the database.

3.1.6.2 LSA Beam Production Chain Information

The LSA system stores additional meta-data about the particular beam production chains. This information shall be transferred to the archiving systems storage.

The data extractor (see chapter 3.1.3) must be able to read this information from the database and correlate it with corresponding archive entries.

3.2 Data Aggregation

The Archiving System should be able to accept data from the following data sources:

- FESA devices via JAPC/RDA subscriptions
- Various Sources via a 'Push API'

A collector component shall be implemented to collect and aggregate data. Its main function is to establish subscriptions to various devices and provide an API to push the data to the system. This component needs to be highly scalable and able to distribute the load over multiple instances of its own.

At least some of the collected data needs to be checked against customisable rules for consistency. This validation should be performed by a separate component.

The rules for the validation must be customisable. For example it can include following:

- Value range suitable for a particular value
- How often a device should send data
- Whether a specific Boolean value may be true (or false) during a particular Beam Mode

Accumulated data needs to be transferred to the database. Therefore a component called DB Writer must be implemented. It must accumulate the data from the collector component instances and write it to the short-term database. Additionally, some of its parts shall handle the information coming from the timing event monitor (see chapter 3.1.6).

All components should be configurable using the Admin GUI.

3.2.1 Data Collector

The Data Collector component is responsible for the actual data acquisition process and communicates directly with the data suppliers. Data should be polled or requested via JAPC subscription.

Since the network bandwidth as well as CPU performance of a single machine is limited, the collector component must be designed to scale linearly in a distributed matter. The amount of instances/threads of collectors needs to be freely configurable.

Additionally, a client side API should be provided to allow the clients to actively push their own data to the Archiving System. The communication protocol between the client side API and the actual aggregation service should be designed in a generic way, allowing other clients to use it. An access control for user-level data sources is required (e.g. using RBAC).

3.2.1.1 FESA and Device Access

Data from the FESA and DevAcc devices should be accessed using JAPC subscriptions.

To protect access to device properties the data collector instances shall be able to utilise the Role Based Access (RBAC) library.

3.2.1.2 Push API

A client-side API should be part of the Aggregation component. This API should allow the clients to archive their own data in the system.

The Push API must be available for both Java (primary) and C++ (secondary) programming languages.

3.2.2 Data Dissemination – Messaging

All data collected by the Data Collector component may also be used by other systems than the Archiving System. The system should implement the data distribution part using a generic messaging framework. A broker based solution fits the requirements of the system at most in this point.

The Apache Kafka [14] product appears to be suitable to fulfil GSI/FAIR requirements and should be aimed for use in the system. Similar to the database options, the to be used messaging broker should be evaluated and tested during the first prototype and release tests. The final choice shall be made as soon as possible after the first prototype has been delivered.

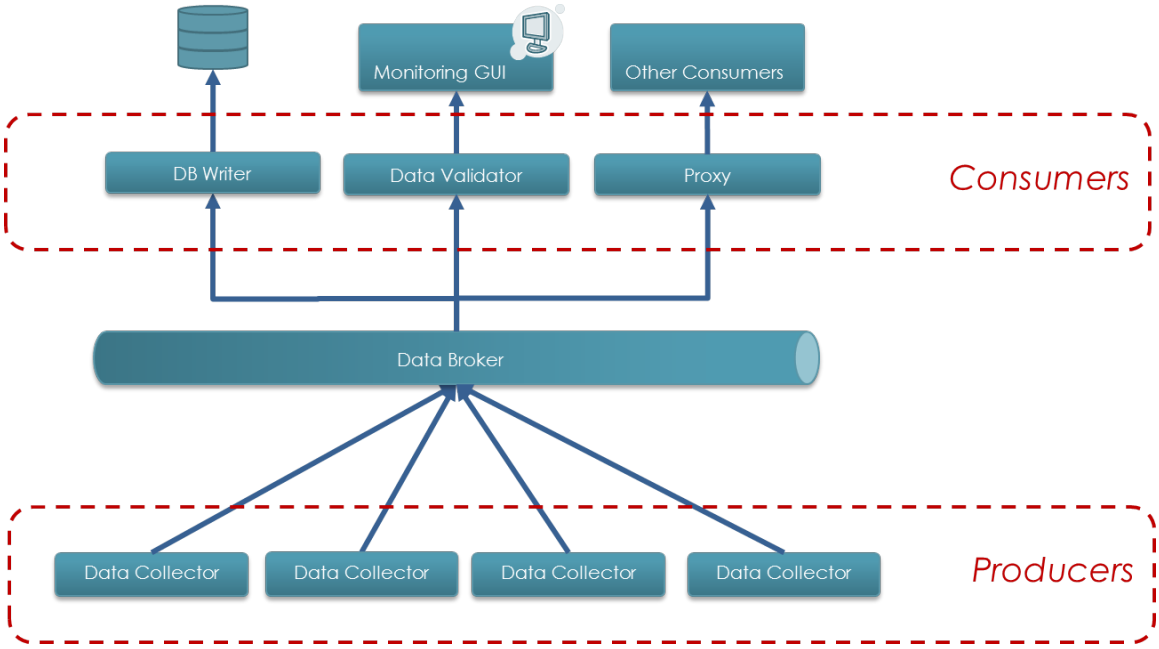

Figure 5: Message Broker Environment

3.2.3 Data Validation

Incoming data must be validated to assure its validity and consistency. Subscriptions must be monitored and re-established if needed to assure that no data to be archived gets lost. Checks should be configurable and are performed by Data Validator component.

The component should be extendible using plug-ins, which means that new rules may be easily added to the system during system operation.

Due to the amount of incoming data, it must be considered if such validation is achievable sequentially before writing into the database. Since the focus of the system is the storing of data it should be realised as a parallel process with an extra output to the GUI. In this case, the errors from validation should also be logged in the archiving database. In some predefined error cases the data written to the database must be marked as potentially corrupt. This note must be presented to the user, in case he requests the data.

The validator component should be connected with the underlying data collector using the messaging broker described in the previous chapter (see Figure 4).

3.2.4 DB Writer

The information collected by multiple collector instances should be accumulated within a component called DB Writer, whose purpose is to optimise and hide the actual writing into the database from the users.

The current design foresees this component being connected to the Apache Kafka Broker, where it collects the data distributed by the collector component and writes it to the database.

The DB Writer implementation must provide a good abstraction level in both directions to the database and to the broker service. This should allow an easy exchange of both components in case of changes to the underlying technology.

In case the database is not accessible, the writer service must be able to temporarily buffer data.

3.2.5 Data Storage Configuration

The Archiving Service must support the user to freely configure which, how often, and how long data should be archived.

'How long' means to specify when an archiving process for a particular parameter should automatically end. The automatic termination of an archiving process should be specified using

- a period or an instant of time
- an event or
- a count of transactions or a size of data stored

The following parameters of the system should be configurable:

- Which parameters from which devices should be archived
- Whether a push or a pull subscription should be used for a particular device/parameter
- How often a particular device should send data
- Logging configuration (level and output)
- DB writing configuration (e.g. how often to commit data, how big the chunks should be)

3.3 User Interface

A graphical user interface (GUI) must be developed as a part of the delivery. The interface itself should consist of multiple independent components.

1. A data display component to request and present the archived data must be provided.
2. A monitoring component must provide monitoring functionality of the Archiving System. It should visualise the results and possible data check errors from the data validation as well as overall system status, performance, warnings, errors, etc.
3. A configuration or admin component should allow the configuration of different components of the archiving system

As it is not yet clear which data displays and correlations will be required in the future, an implementation, which facilitates a generic way to easily add new data displays to the GUI, shall be investigated.

The GUI(s) shall be designed in accordance with the main contractor who ensures close cooperation with the accelerator users, experts and in particular the accelerator Operations group. The GUI(s) must adhere to the GUI Guidelines [9].

3.3.1 Data Graphical User Interface

The Data GUI shall be used to query and present the data to the user. As with other GUIs this component shall be implemented as a standalone application and utilise the Client Extraction API, as described before. A user should be able to select specific physics data parameters for a given period of time or for a given beam production chain. The GUI should also provide functionality to find a particular beam production chain with desired parameters e.g. beam on/off, a particular element etc.

4 Quality Assurance, Tests and Acceptance

The system to be built must adhere to the guidelines and recommendations for software developments in the FAIR accelerator control system context, as referenced in the FAIR Common Specification F-CS-C-01e (Common Specification Accelerator Control System). The supplier of the work package must identify the relevant standards and recommendations before start of the development. Details must be fixed as part of the technical design concept in the initialization phase.

4.1 Development Methodology

The Archiving System shall be developed in an iterative and incremental methodology.

Each iteration cycle shall result in a running system, which can be evaluated and tested at the FAIR site. The first iteration, which has to be available as early as possible, must concentrate on the most critical functionality. In successive iterations, the system is enhanced by adding features until the desired total functionality is reached.

In the initial phase, the technical design concept and the plan for the iterations must be developed, and approved by the FAIR contracting body. At the end of each iteration cycle the achieved status of the system will be evaluated and the iteration plan – if necessary – adjusted. Each iteration cycle must be approved by the FAIR contracting body before it can be started.

4.2 Quality Assurance System of the Supplier

Generally, the software specific measures of Quality Assurance described in Common Specification 'Accelerator Control System' [2] fully apply.

4.3 FAT & SAT

The Common Specification 'Accelerator Control System' [2] fully applies.

5 Documentation

The Common Specification 'Accelerator Control System' [2] fully applies.

6 Warranty

The conditions and warranty period specified in the contract applies.

7 Scope of Delivery

Following components have to be delivered:

Name	Short description
Technical concept (CDR, FDR documents)	Describes the technical details of proposed solution especially the implemented interfaces (conceptual design report, final design report).
Data Collector	See chapter 3.2.1
Data Model	Formal description of the model used to store the data in the database. This includes the specification of both: the STS and the LTS models.
Configuration specification	If the running of the system requires some specific configurations of the services (database, broker), not covered by a normal delivery, then the specification of such configuration must be delivered
Data Writer component	See chapter 3.2.4
Data Extractor component	See chapter 3.1.3
Client API for extraction service	A client side API to access the interface described in chapter 3.1.3
Data Reduction components	Depending on final technical choice this part may be implemented differently and hence may consist of different components. Presented solutions either contain a database job, aggregating and transferring the data from one source to another, or a specific DB writer, which aggregates and writes massages to LTS in parallel. See chapter 3.1.1
Data Validation service	See chapter 3.2.3
GUI application for monitoring and configuration	A GUI to monitor the basic characteristics of the Archiving System and configure the archiving system. These may be implemented as two independent applications.
GUI application for data aggregation	A GUI to access and display the data stored in the archiving database.

Table 3: Deliverables

I. Related Documents

[1] “List of Abbreviations for Controls”, 2012-09-06, EDMS #1235811 v.1

[2] F-CS-C-01e, FAIR Common Specification 'Accelerator Control System', EDMS #1174186 v.6

[3] F-DS-C-03e, FAIR Detailed Specification 'Settings Management System', EDMS #1176027 v.1

[4] F-DS-C-09e, FAIR Detailed Specification 'Alarm System', EDMS #1176037 v.4

[5] F-DS-C-12e, FAIR Detailed Specification 'Beam Transmission Monitor System', EDMS #1176040 v.4

[6] F-TG-C-02e, Technical Guideline – ACOS Equipment Interfaces, EDMS #1171994 v.2

[7] F-DS-C-15e, FAIR Detailed Specification 'FEC Device Classes (FESA)', EDMS #1176043 v.3

[8] F-DS-C-13e, FAIR Detailed Specification 'Post Mortem System', EDMS #1176041 v.5

[9] F-DG-C-02e, FAIR Control System GUI Development Guideline, EDMS #1235719 v.1

[10] F-DG-C-03e, FAIR Control System Software Architecture Guideline, EDMS #1235720 v.1

[11] A Survey of Basic Storage Systems for Large Amounts of Particle Accelerator Data, Winfried Schütte, DESY MST, PCaPAC 2000

[12] F-DG-C-01e, FAIR Control System FESA Development Guideline, EDMS #1235310 v.2

[13] Apache Cassandra website: http://cassandra.apache.org/

[14] Apache Kafka website: http://kafka.apache.org/

[15] Apache Spark website: http://spark.apache.org/

[16] “FAIR Diagnostic Logging” Technical Design and Implementation 2012

[17] Elasticsearch website: https://www.elastic.co/de/products/elasticsearch