I. PROJECT IDENTIFICATION

a. Award ID Number:	DE-SC0022583
b. Agency:	US Department of Energy, Office of High Energy Physics
c. Award Program:	SBIR Initial Phase II, DE-FOA-0002991
Topic:	29 (d): Data Infrastructure for the Next Generation of Adaptive Real-Time Controls for Large-Scale Facilities
d. Project Title:	A Data Science and Machine Learning Platform Supporting Large Particle Accelerator Control and Diagnostics Applications
e. Principal Investigator:	Leo R. Dalesio bdalesio@ospreydcs.com (443) 834-3775
i. Company Information:	Osprey Distributed Control Systems 304 Blue Heron Court Ocean City, Maryland 21842-2452
j. Grant Period:	Aug 2023 to July 2025
k. Reporting Period:	Nov 21, 2023, to March 19, 2023
l. Report Term:	Quarter 2
Proprietary Data:	There is no proprietary data. All work performed in this project is open source. It will be developed in a GIT repository and provided to the EPICS collaboration for use throughout that community.

Table of Contents

I. PROJECT IDENTIFICATION 1
II. ACCOMPLISHMENTS 4
1. PROJECT GOALS 4
2. QUARTER ACCOMPLISHMENTS 5
i. Query Service 6
ii. Annotations Service 9
iii. Administration API Library Components 10
iv. Java Language API Library 11
v. Web Application Development 13
vi. Documentation 15
3. TRAINING OPPORTUNITIES AND PROFESSIONAL DEVELOPMENT 15
4. DISSEMINATION OF PROJECT RESULTS 15
5. ACTIVITY SCHEDULE FOR NEXT QUARTER 15
II. PRODUCTS 17
1. PUBLICATIONS 17
2. INTELLECTUAL PROPERTY 17
3. TECHNOLOGIES AND TECHNIQUES 17
4. OTHER PRODUCTS 18
III. PARTICIPANTS AND COLLABORATING ORGANIZATIONS 18
1. PARTICIPANTS 18
2. PARTNERS 19
3. OTHER COLLABORATORS 19
IV. IMPACT 19
1. IMPACT ON PRINCIPLE DISCIPLINE 20
2. IMPACT ON OTHER DISCIPLINES 20
3. IMPACT ON DEVELOPMENT OF HUMAN RESOURCES 20
4. IMPACT ON PHYSICAL, INSTITUTIONAL, AND INFORMATION RESOURCES 20

5. IMPACT ON TECHNOLOGY TRANSFER 20
6. IMPACT ON SOCIETY BEYOND SCIENCE AND TECHNOLOGY 20
7. FOREIGN SPENDING 20

V. CHANGES AND PROBLEMS 20
1. CHANGES IN APPROACH 21
2. ACTUAL AND/OR ANTICIPATED PROBLEMS 21
3. CHANGES REQUIRING SIGNIFICANT IMPACT ON EXPENDITURE 21
4. SIGNIFICANT CHANGES IN USE OF HUMAN SUBJECT, ANIMALS, AND/OR BIOHAZARDS 21
5. CHANGES IN PRIMARY PERFORMANCE SITE LOCATION 21
6. CARRYOVER AMOUNT 21

VI. DEMOGRAPHIC INFORMATION 21
VII. SPECIAL REPORTING REQUIREMENTS 21
REFERENCES 22

Research Performance Progress Report: Quarter 2

DE-SC0022583 SBIR Initial Phase II

II. ACCOMPLISHMENTS

The focus of the Quarter 2 activities was to build out the Machine Learning Data Platform (MLDP) core services according to the new system design introduced in Quarter 1. Specific attention was on the Query Service although multiple activities were initiated as described below. Recall that the Year 1 performance milestone of 100 Mbytes/second[1] (Mbps) data rates was achieved in Quarter 1 [1]. This achievement required a full redesign of the MLDP core systems affecting all core services such as data ingestion, data archiving, and query services. Most core components of the original prototype MLDP have been abandoned in favor of the new high-performance design.

1. Project Goals

For convenience of reference, we include the following tables providing itemized task lists and milestones as stated in the SBIR Phase II project narrative. The tables also include estimated schedules, levels of effort, and staffing.

Year 1

Task	Items	Start	End	Effort	Yearly	Milestones
Datastore Upgrade	Upgrade/Redesign	Q1	Q3	6 MM		100 Mbyte/sec Full Annotations
	Performance	Q1	Q3	3 MM		
	Annotations	Q2	Q3	1 MM		
	Senior Staff			10 MM	0.83 FTE	
APIs	Administration API	Q2	Q3	2 MM		
	Python Query API	Q3	Q3	1 MM		
	Junior/Senior Staff			3 MM	0.25 FTE	
Adv Data Science	Applications	Q2	Cont.	1 MM		
	Archive Support	Q4	Q4	2 MM		
	Senior Staff			3 MM	0.25 FTE	
Deployment	MLDP Testing	Q4	Q4	1 MM		Data Throughput Aggregator Install.
	Aggregator Install.	Q4	Cont.	1 MM		
	Senior Staff			2 MM	0.16 FTE	
Web Application	Upgrades	Q1	Cont.	2 MM		
	Visualization	Q1	Cont.	1 MM		
	Data Science	Q1	Cont.	1 MM		
	Junior/Senior Staff			4 MM	0.33 FTE	

Table 1: Phase II Year 1 Performance Schedule

[1] The original proposal stated rates of 32 Mbps for native 8-byte wide double values and 100 Mbps for Java 24-byte wide Double objects. For the sake of clarity Osprey DCS has standardized all performance criteria to consider only native formats in conformance with low-level transport and storage mechanisms.

Year 2

Task	Items	Start	End	Effort	Yearly	Milestones
Adv Data Science	Applications	Q1	Q4.	2 MM		MLDP Appl. Plugin Operation
	Datastream Process.	Q1	Q3	4 MM		
	Plugin Framework	Q1	Q3	4 MM		
	Integrat./Performance	Q2	Q4	4 MM		
	Senior Staff			14 MM	1.16 FTE	
APIs	ADS Support	Q2	Q3	1 MM		Facility Install.
	Documentation	Q3	Q3	1 MM		
	Senior Staff			2 MM	0.16 FTE	
Deployment	Deployment Systems	Q1	Q4	1 MM		Cloud Deploy. Facility Install.
	Cloud Deployment	Q1	Q4	1 MM		
	Facility Installation	Q1	Q4	1 MM		
	Senior Staff			3 MM	0.25 FTE	
Web Application	Commercialization	Q1	Q4	2 MM		Remote Demon.
	Data Science	Q1	Q4	1 MM		
	Junior/Senior Staff			3 MM	0.25 FTE	

Table 2: Phase II Year 2 Performance Schedule

2. Quarter Accomplishments

The Ingestion Service was central to Quarter 1 research and development since it was crucial to the performance objective: the MLDP must ingest and archive data at the stated rate. Quarter 2 focused on developing the complementary Query Service in accordance with the new archive structure and schema. Crucial to the Query Service is the design of a robust, well-defined interface capable of supporting all stated data science and machine-learning applications.

Additionally other project tasks were initiated in Quarter 2, most according to the original project performance schedule listed in Table 1 and predicted in the Quarter 1 Report, Activity Schedule for Quarter 2 [1]. The Quarter 2 focus consisted of the following activities:

i. Development of the Query Service core service.

ii. Development of the Annotations Service has been initiated.

iii. Components of the Administration API library have been developed.

iv. Rebuilding the Java language API library conforming to new communications framework.

v. Development of the Web Application has been initiated.

Additionally, the overall structure and robustness of the MLDP core systems continues to be refactored and improved. The entire core system has been reduced to 3 repositories: 1) the (gRPC) communications framework, 2) the core services implementation (i.e., the *Data Platform*), and 3) the installation and deployment system.

Note that work on the MLDP installation system was started early, in Quarter 1. The system saw continued improved and expansion throughout Quarter 2; it was found that the ability to rapidly deploy the latest operational release provided significant development benefits. Team members (and interested external parties) can evaluate and utilize the latest development builds and resouces without having to maintain and update local builds and configurations.

Before proceeding to the Quarter 2 accomplishments, the basic architecture of the Machine Learning Data Platform (MLDP), including the new core services design, is shown in Figure 1. This diagram was provided in the Quarter 1 report but is repeated here for convenience of reference. The diagram provides a concise depiction of the MLDP architecture, deployment, and operation, which is useful for below reference.

Recall that we denote the MLDP core services collectively as the *Data Platform*. As seen in Figure 1 the new Data Platform architecture is composed of a set of collaborating services, all communicating through a gRPC framework (i.e., the *communications framework*). The new design is simplified, performance based, and distributed. The diagram also identifies the MLDP subsystem and client relationships.

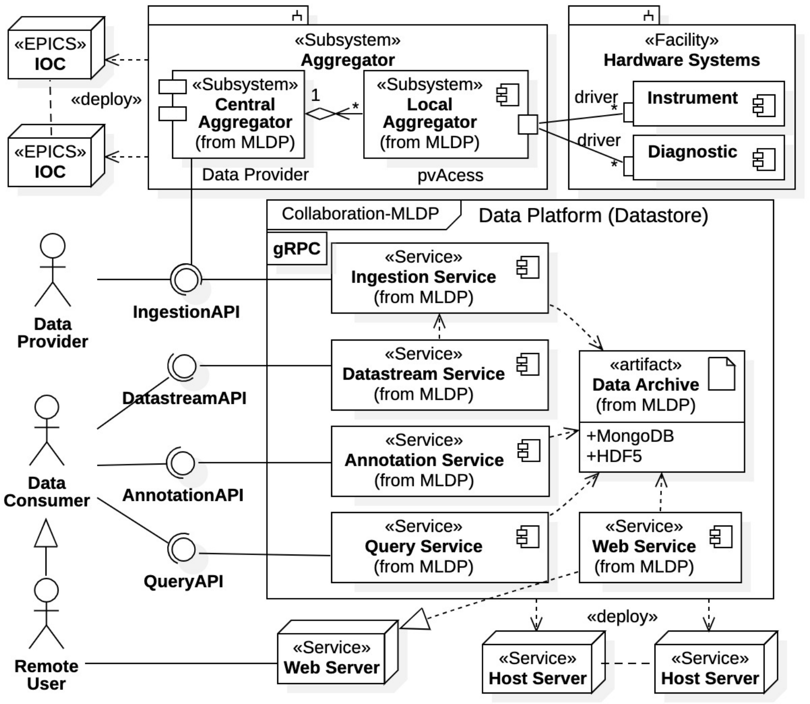

«EPICS» IOC
«EPICS» IOC
«deploy»
«Subsystem» Aggregator
«Subsystem» Central Aggregator (from MLDP)
«Subsystem» Local Aggregator (from MLDP)
1
*
Data Provider
pvAcess
«Facility» Hardware Systems
driver
Instrument
driver
Diagnostic
Collaboration-MLDP
Data Platform (Datastore)
gRPC
«Service» Ingestion Service (from MLDP)
«Service» Datastream Service (from MLDP)
«Service» Annotation Service (from MLDP)
«Service» Query Service (from MLDP)
«Service» Web Service (from MLDP)
«artifact» Data Archive (from MLDP)
+MongoDB
+HDF5
Data Provider
IngestionAPI
DatastreamAPI
Data Consumer
AnnotationAPI
QueryAPI
Remote User
«Service» Web Server
«deploy»
«Service» Host Server
«Service» Host Server

Figure 1: MLDP composite diagram with client relationships

i. Query Service

The basic structure and fundamental operations of the Query Service was completed in Quarter 2. The new Query Service is now included in the MLDP installation system. Included in the deployment are scripts for starting and stopping the Query Service along with an application for testing and benchmarking its operation on local platforms. Query validity and data integrity has been verified using components of the Administration API also completed in Quarter 2 (described further in the Subsection I.2.iii below). Additionally, the gRPC communications framework was narrowed and simplified.

The most important factor concerning the Query Service is its functionality; this is the aspect of the MLDP seen by end users (the Data Consumer of Figure 1). Thus, the Query Service must support all general data science and machine learning applications supported by the project. The performance of the Query Service is not as critical as that for the Ingestion Service; however, our objective is to have Query Service performance on par with the Ingestion Service (data rates of at least 32 Mbps). The Ingestion Service was redesigned and rebuild in Quarter 1, which required a redesign of the MLDP data archive and communications framework. Consequently, the Query Service required a complete rebuild to conform with the new archive and gRPC communications framework.

To appreciate the end-user requirements of the Query Service a use case diagram is provided in Figure 2. In the diagram the end-user has the general label *Data Consumer*, which is any party interested in the data archive, such as data scientists, engineers, operators, and software applications. As seen in the figure, the Query Service supports most general data analysis applications for the MLDP.

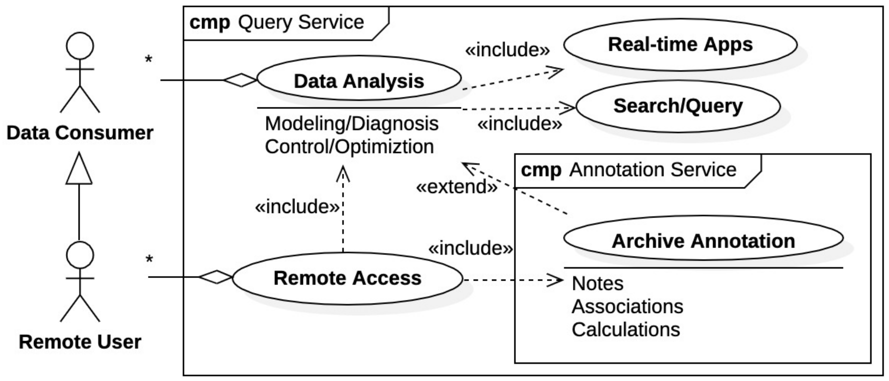
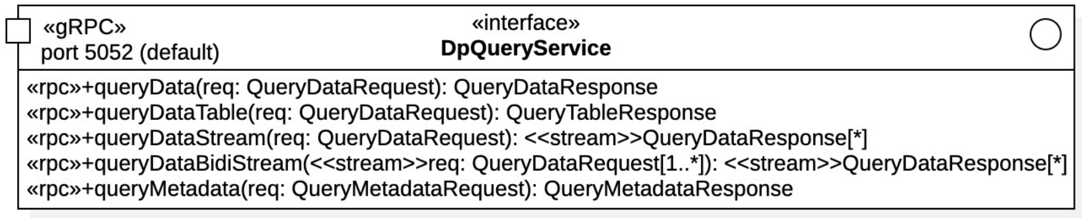


Figure 2: Query Service use-case diagram

Figure 2 also includes a Remote User which is a subclass of Data Consumer. Remote access to the MLDP archive is supported by the Web Application discussed in Subsection I.2.v below. It provides a subset of Query Service operations which are available to all remote locations via a standard internet web browser. Also shown in the figure is the Annotation Service which allows Data Consumers to annotate the data archive with notes, data associations, and post-acquisition calculations. In the new design the Annotation Service is a fully independent core service although, as seen in the diagram, there are shared aspects with the Query Service.

From MLDP prototype evaluations in Phase I it was found that advanced data processing within the Query Service creates significant performance costs for all active clients. The new design facilitates distributed data processing between the Query Service and its clients. This option is provided explicitly in the new Query Service communications interface shown in Figure 3.


«gRPC»
port 5052 (default)
«interface»
DpQueryService

```
«rpc»+queryData(req: QueryDataRequest): QueryDataResponse
«rpc»+queryDataTable(req: QueryDataRequest): QueryTableResponse
«rpc»+queryDataStream(req: QueryDataRequest): <<stream>>QueryDataResponse[*]
«rpc»+queryDataBidiStream(<<stream>>req: QueryDataRequest[1..*]): <<stream>>QueryDataResponse[*]
«rpc»+queryMetadata(req: QueryMetadataRequest): QueryMetadataResponse
```

Figure 3: Query Service gRPC communications framework interface

The communications interface for the new Query Service depicted in Figure 3 illustrates the robust nature of the new design; all required functionality of the Query Service is supported by a narrow interface. There are currently 2 classes of queries supported by the Query Service:

1. Time-series data queries: These operations are prefixed with `queryData…` in Figure 3 and require a `QueryDataRequest` message to initiate. They return heterogeneous, correlated, time-series data in both tabular format (operation `queryDataTable()`) and raw data format (all other `queryData…` operations).
2. Metadata queries: This is the single operation `queryMetadata()` which requires a `QueryMetadataRequest` message to initiate. Metadata queries return information about the MLDP data archive itself. Typically, metadata query results are used by clients to refine time-series data queries. The current Query Service development release supports metadata concerning data sources (i.e., "process variables") that have contributed to the archive. Eventually all metadata information acquired during the ingestion process will be available.

The different forms of the time-series data queries (i.e., operations prefixed with `queryData…`) are provided for either convenience or performance. Clearly the `queryDataTable()` operation is the most convenient, returning results in tabular form. However, it is not the fastest nor does it support all the features of the Query Service. Clients desiring maximum performance are provided the option of streaming raw time-series query results with operations `queryDataStream()` and `queryDataBidiStream()`; these are the fastest possible data transfer options. However, the burden of data reconstruction is then left to the client, a design decision enabling distributed data processing. Because the Query Service can concurrently service multiple clients, best performance delegates that some data processing be offloaded to the clients. Otherwise, data processing wait times for a single client are shared among all clients. The Java language API library rebuild initiated this quarter performs the time-series data reconstruction on the client side, thus, unburdening the Query Service of these processing requirements.

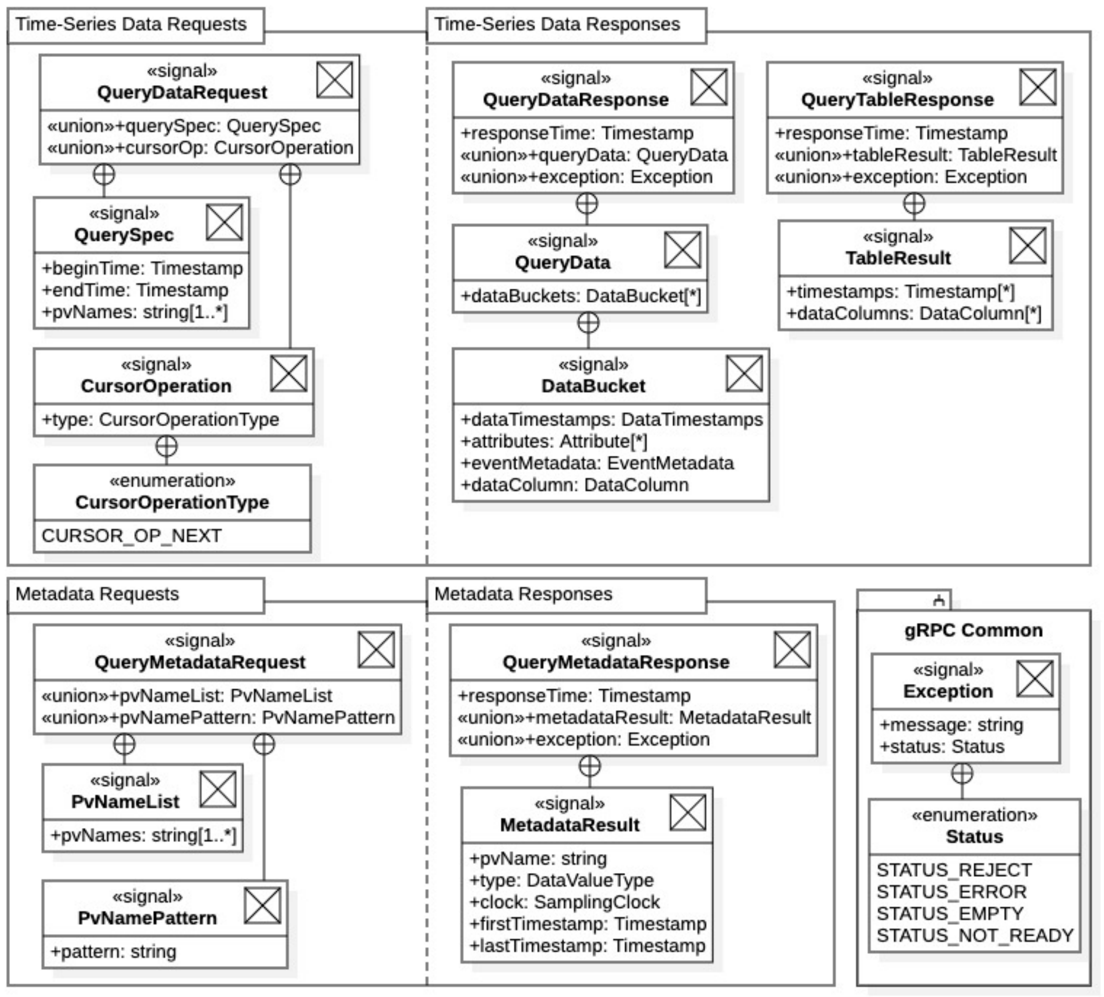

Time-Series Data Requests
«signal» QueryDataRequest
«union»+querySpec: QuerySpec
«union»+cursorOp: CursorOperation
«signal» QuerySpec
+beginTime: Timestamp
+endTime: Timestamp
+pvNames: string[1..*]
«signal» CursorOperation
+type: CursorOperationType
«enumeration» CursorOperationType
CURSOR_OP_NEXT
Time-Series Data Responses
«signal» QueryDataResponse
+responseTime: Timestamp
«union»+queryData: QueryData
«union»+exception: Exception
«signal» QueryData
+dataBuckets: DataBucket[*]
«signal» DataBucket
+dataTimestamps: DataTimestamps
+attributes: Attribute[*]
+eventMetadata: EventMetadata
+dataColumn: DataColumn
«signal» QueryTableResponse
+responseTime: Timestamp
«union»+tableResult: TableResult
«union»+exception: Exception
«signal» TableResult
+timestamps: Timestamp[*]
+dataColumns: DataColumn[*]
Metadata Requests
«signal» QueryMetadataRequest
«union»+pvNameList: PvNameList
«union»+pvNamePattern: PvNamePattern
«signal» PvNameList
+pvNames: string[1..*]
«signal» PvNamePattern
+pattern: string
Metadata Responses
«signal» QueryMetadataResponse
+responseTime: Timestamp
«union»+metadataResult: MetadataResult
«union»+exception: Exception
«signal» MetadataResult
+pvName: string
+type: DataValueType
+clock: SamplingClock
+firstTimestamp: Timestamp
+lastTimestamp: Timestamp
gRPC Common
«signal» Exception
+message: string
+status: Status
«enumeration» Status
STATUS_REJECT
STATUS_ERROR
STATUS_EMPTY
STATUS_NOT_READY

Figure 4: Query Service communications framework messages

The ability to support the broad search and query requirements of the Query Service completely within the narrow interface of Figure 3 is facilitated by the design of the `QueryDataRequest` and `QueryMetadataRequest` request messages and their respective response messages `QueryDataResponse` and `QueryMetadataResponse`. An abbreviated description of this design is shown in Figure 4. The basic structure is presented to demonstrate how the communications framework supports the Query Service functionality.

All time-series data requests are initiated with a `QuerySpec` message within a `QueryDataRequest`, which identifies the request. If a table request operation is initiated, then the response is a `QueryTableResponse` message containing an ordered list of timestamps and all data columns matching the request (note data columns may be of different data types). If a raw data request is performed, the response is a stream of `QueryDataResponse` messages all with `QueryData` messages containing all requested data in the form of unformatted "data buckets". It is important to note that since a gRPC data stream is used to transport the raw time-series data requests, the responses are essentially unlimited in size. Specifically, result sets are not subject to gRPC message size limitations (typically 4-16 Mbytes). In principle, a single, raw time-series data request could stream the entire MLDP data archive back to the client. For additional performance the client can established multiple concurrent data streams to recover a time-series data request; thus, data transport performance is essentially determined by the host network capability.

Metadata requests are all supported by the `QueryMetadataRequest` message. It contains a union of possible metadata request types, the response depending upon the specific request invoked. As indicated in Figure 4, currently only data source metadata queries are available (i.e., "PV requests"). Other types of metadata requests are added to the Query Service in the same fashion, by added metadata request types to the union.

Finally, note that all Query Service response messages contain *either* the requested data or metadata, *or* an `Exception` message. This is the general mechanism for exception handling within the MLDP core services. Clients must always check for exceptions in their request responses; the details of any request errors are contained in the `Exception` message.

In summary, the Query Service is now operational and is included within the MLDP development installation. It has demonstrated basic functionality but is not fully featured yet. The metadata request has basic design and functionality is not built out. Additionally, only scalar data types are available in the time-series data archive, structured data has not yet been included.

ii. Annotations Service

Development of the Annotations Service was initiated in Quarter 2. The Annotations Service allows MLDP clients (i.e., Data Consumers) to annotate the data archive with comments, data associates, post-acquisition calculations, and other artifacts. These annotations are then available for other MLDP users to inspect, analyze, and contribute additional annotations. The annotations feature of the MLDP allows Data Consumers to create "value-added" content within the data archive which is then available to all other users.

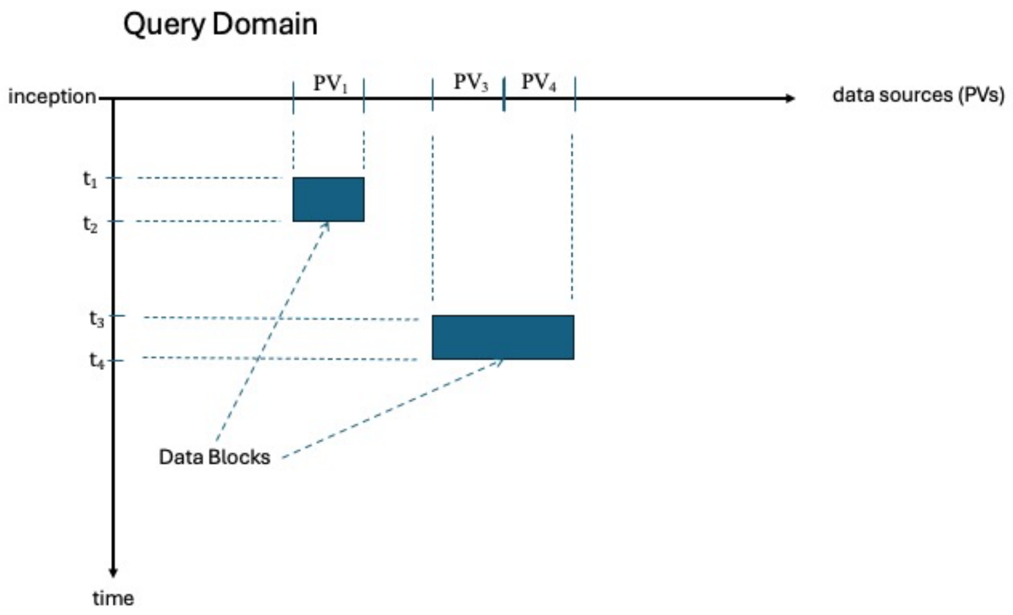


Figure 5: the MLDP data archive query domain

The Annotations Service is a sophisticated component of the MLDP requiring a significant design phase. The first step in design was defining an *annotation* and exactly that which is annotated (i.e., the annotation *target*). Our approach is shown in the conceptual diagram of Figure 5. There the entire data archive is depicted as a quadrant of a 2-dimensional *query domain* with the axes of *time* and *data source*. The notion is to define a collection of well-defined "basis sets" that will cover the entire query domain. Such basis sets are depicted as the Data Blocks within the figure. Each *data block* is composed of a time interval $[t_i, t_{i+1}]$ and a set of data sources $\{PV_j, PV_{j+1}, PV_{j+2}, \ldots\}$; note that the data sources need not be contiguous. Thus, any data block is described as the set $[t_i, t_{i+1}]\times\{PV_j, PV_{j+1}, PV_{j+2}, \ldots\}$. Clearly the data blocks form a basis which covers the entire query domain, and we can use collections of data blocks to identify any target of any archive annotation.

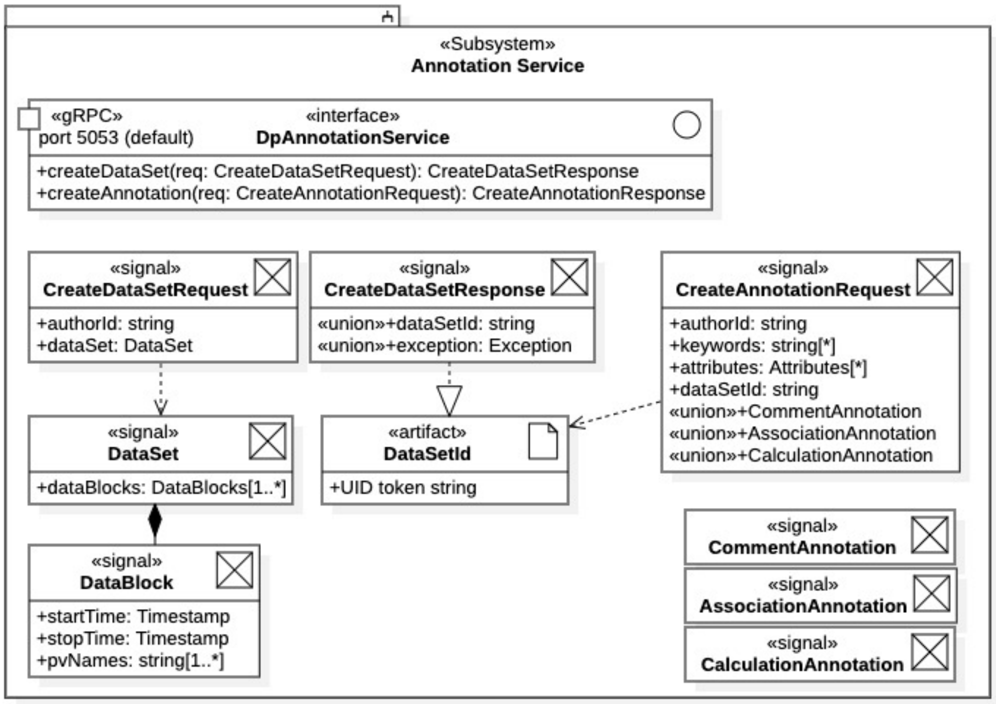


Figure 6: basic Annotation Service gRPC interface (communications framework)

The design of the Annotation Service formalizes the above concepts. These ideas are captured in the gRPC interface schematic shown in Figure 6. An annotation applies to any collection of data blocks in the query domain, which we denote a *data set* and represent with the `DataSet` message. The definition of data set is so essential to the operation of the Annotation Service that we have made its creation an explicit operation. Referring to Figure 6, the creation of an archive annotation is thus a 2-step process: 1) the targeted archive domain is first identified with operation `createDataSet()`, then 2) the annotation itself is created with operation `createAnnotation()` using a generated data set identifier. Note that when creating an annotation with a client language API library or with the Web Application this two-step process is generally hidden from users.

There is additional value to the explicit definition and creation of data sets. By its nature a data set defines a region of the query domain and, thus, *is* a time-series data request. We are currently exploring the utility of this approach across the MLDP; specifically, this could be a universal approach for both the Query Service and the Annotation Service.

iii. Administration API Library Components

The Administration API library provides a collection of tools for managing MLDP deployments. It is to provide tools and applications for the following tasks:

1. Testing availability and connectivity of all MLDP core services in all possible configurations.
2. Performance testing and benchmarking of core services.
    a. Ingestion Service verification and performance benchmarking.
    b. Query Service verification and performance benchmarking.
    c. Annotation Service verification and validation.
3. Managing the MDLP data archive.
    a. Data validation.
    b. Corruption detection.
    c. Data throughput verification and integrity.
    d. Compacting, purging, and/or cold storage of legacy data.

In Quarter 2 a focused effort was made to create a verification and benchmarking system for the Query Service. As a consequence, a generalized verification and benchmarking system was developed for both the Ingestion Service and the Query Service. Additionally, within the Query Service development process, it was recognized that a formal system for validating full data throughput was needed; specifically, we required a tool for verifying that any data stored in the archive via the Ingestion Service is exactly that data recovered via the Query Service. Thus, a formal tool for data throughput validation was also built during Quarter 2.

Although creation of the above tools required significant developer resources in Quarter 2, the task greatly reduced overall development efforts. With the installation and deployment system now in place, whenever a new Query Service development release is available, all team members can download it, install it, and immediately begin using it. Data integrity tests are available as unit tests within the installation which can immediately verify coordinated operation of both the Ingestion Service and the Query Service on local platforms. Additionally, the benchmarking applications are available for comparisons and tuning of local installations (note that both services have configuration parameters that can be tuned for specific platforms).

During Quarter 2 several upgrades and modifications were made to the gRPC communications framework. We wish to acknowledge that availability of the above tools was instrumental during this process. The design of the gRPC framework and API is critical to the MLDP and extensive effort has been devoted to ensuring the robustness and flexibility of this communication system. The ability to make upgrades then immediately verify their operations was essential.

Currently the above tools reside in the unit testing libraries of the core services and as independent applications that ship with the MLDP installation. The intent is to eventually create a client front-end GUI application for the entire suite of administration applications and tools, which we denote the Administration API. The format of the Administration API is still under consideration (e.g., standalone application or internet browser application).

iv. Java Language API Library

With the availability of an operational Query Service development release, development was initiated for the Java language API client library. This library was available for the MLDP prototype, however, due to the extensive redesign and rebuilding of the core services the library required substantial refactoring for a viable port. It was decided that a completed rebuild was appropriate, although leveraging off the existing prototype library code base. The intent is to keep the same basic client facing presentation, but with simplifications and greater “usability”, while conforming to the new communications interface and core services.

Within the Java client API library, the generalized gRPC connectivity and communication mechanism with the MDLP core services has been established (although essentially hidden from clients). As with the original Java client library, the core services connectivity is presented to clients in the form of "connection factories" as described in the previous MLDP documentation [2]. Connection factories allow clients to connect to services using optional parameters, including all default values specified in the library configuration. The Query Service API within the Java client library has been recently completed and is currently undergoing testing.

The basic structure of the Query Service interface in the Java API library is shown in Figure 7. All client interaction with the Query Service is facilitated through an `IDpQueryService` interface. Interface instances are obtained via the connection factory `DpQueryServiceFactory`. As seen in the diagram, the connection factory offers various options for configuring `IDpQueryService` interface connections, including the default connection operation `connect()`. Note that a single client can create multiple interfaces to enable concurrency and multi-threading. Whenever a client is finished using an interface it should be explicitly shutdown to release all associated communications resources maintaining overall performance.

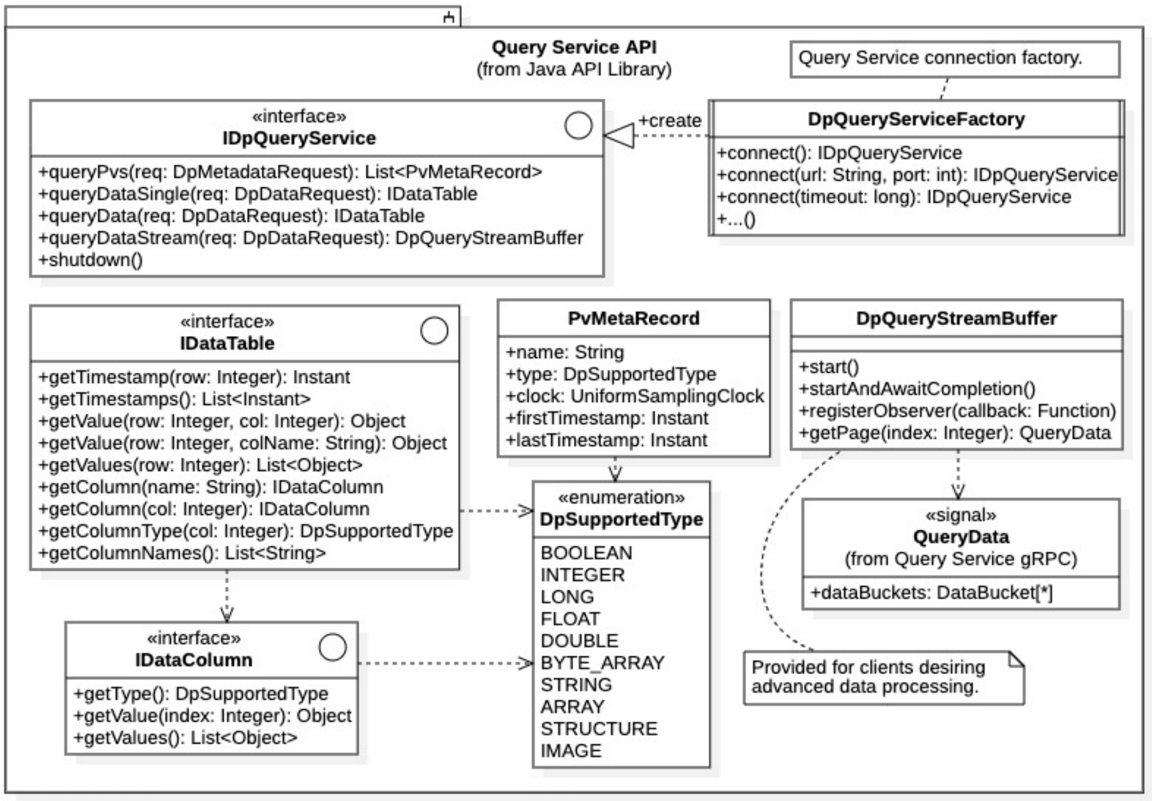

Query Service API
(from Java API Library)
Query Service connection factory.
«interface»
IDpQueryService
+queryPvs(req: DpMetadataRequest): List<PvMetaRecord>
+queryDataSingle(req: DpDataRequest): IDataTable
+queryData(req: DpDataRequest): IDataTable
+queryDataStream(req: DpDataRequest): DpQueryStreamBuffer
+shutdown()
+create
DpQueryServiceFactory
+connect(): IDpQueryService
+connect(url: String, port: int): IDpQueryService
+connect(timeout: long): IDpQueryService
+...()
«interface»
IDataTable
+getTimestamp(row: Integer): Instant
+getTimestamps(): List<Instant>
+getValue(row: Integer, col: Integer): Object
+getValue(row: Integer, colName: String): Object
+getValues(row: Integer): List<Object>
+getColumn(name: String): IDataColumn
+getColumn(col: Integer): IDataColumn
+getColumnType(col: Integer): DpSupportedType
+getColumnNames(): List<String>
PvMetaRecord
+name: String
+type: DpSupportedType
+clock: UniformSamplingClock
+firstTimestamp: Instant
+lastTimestamp: Instant
DpQueryStreamBuffer
+start()
+startAndAwaitCompletion()
+registerObserver(callback: Function)
+getPage(index: Integer): QueryData
«enumeration»
DpSupportedType
BOOLEAN
INTEGER
LONG
FLOAT
DOUBLE
BYTE_ARRAY
STRING
ARRAY
STRUCTURE
IMAGE
«signal»
QueryData
(from Query Service gRPC)
+dataBuckets: DataBucket[*]
«interface»
IDataColumn
+getType(): DpSupportedType
+getValue(index: Integer): Object
+getValues(): List<Object>
Provided for clients desiring advanced data processing.

Figure 7: Query Service interface within Java language API library

The primary time-series query operation is given by `queryData()`, which requires a `DpDataRequest` object (not shown) that defines the request. The `DpDataRequest` class functions as a request builder allowing clients to configure data request with methods for selecting data sources, time ranges, attributes, and filtering. The result of a time-series data request is a data table object exposing the `IDataTable` interface. It is desirable to hide the table implementation details from clients as multiple implementations are available (the library selects which is appropriate). Shown in Figure 7 are the various expected operations for table access within interface `IDataTable`, including by entire table column which is represented generally as a `IDataColumn` interface implementation.

Note that the `queryDataStream()` operation is also available for time-series data requests; this option allows advanced clients to do their own dynamic data processing when desired. A `DpQueryStreamBuffer` object is returned which gives clients access to the dynamic gRPC data stream and the raw Protocol Buffers messages it manages.

The metadata queries within `IDpQueryService` are available by metadata type. Currently the Query Service supports only metadata for data source information (i.e., "Process Variable" or PV information) and this information is available via the `queryPvs()` operation. Metadata queries always return collections of metadata records, in this case `PvMetaRecord` objects. Specifically, the `queryPvs()` operation returns a list of `PvMetaRecord` records, one record for each process variable matching the request.

Currently, the Query Service interface within the Java language API library is functional but not fully tested. It has demonstrated operation with the current MLDP Query Service development release. Additional features will be implemented as they become available within the Query Service. However, when testing is complete the intent is to proceed immediately to Ingestion Service interface development within the library. This action is warranted by the recent opportunity to use the MDLP data archive for fast data-acquisition applications.

v. Web Application Development

Phase II Web Application development was initiated in Quarter 2. A prototype Web Application was built in Phase I to demonstrate proof of principle. However, due to the redesign of the MDLP communications framework and core services, and due to the rudimentary nature of the prototype, we have decided to completely rebuild this component (while leveraging off the prototype system code base).

The Web Application provides remote access and interaction with the MLDP data archive, as shown in the Remote Access use case in Figure 2. It allows remote users universal access to the data archive using a standard internet web browser. The Web Application is an independent system and can be hosted on an independent Web Server platform as shown in the MLDP component diagram of Figure 1. It communicates with the MLDP core services using gRPC through a proxy service (described below).

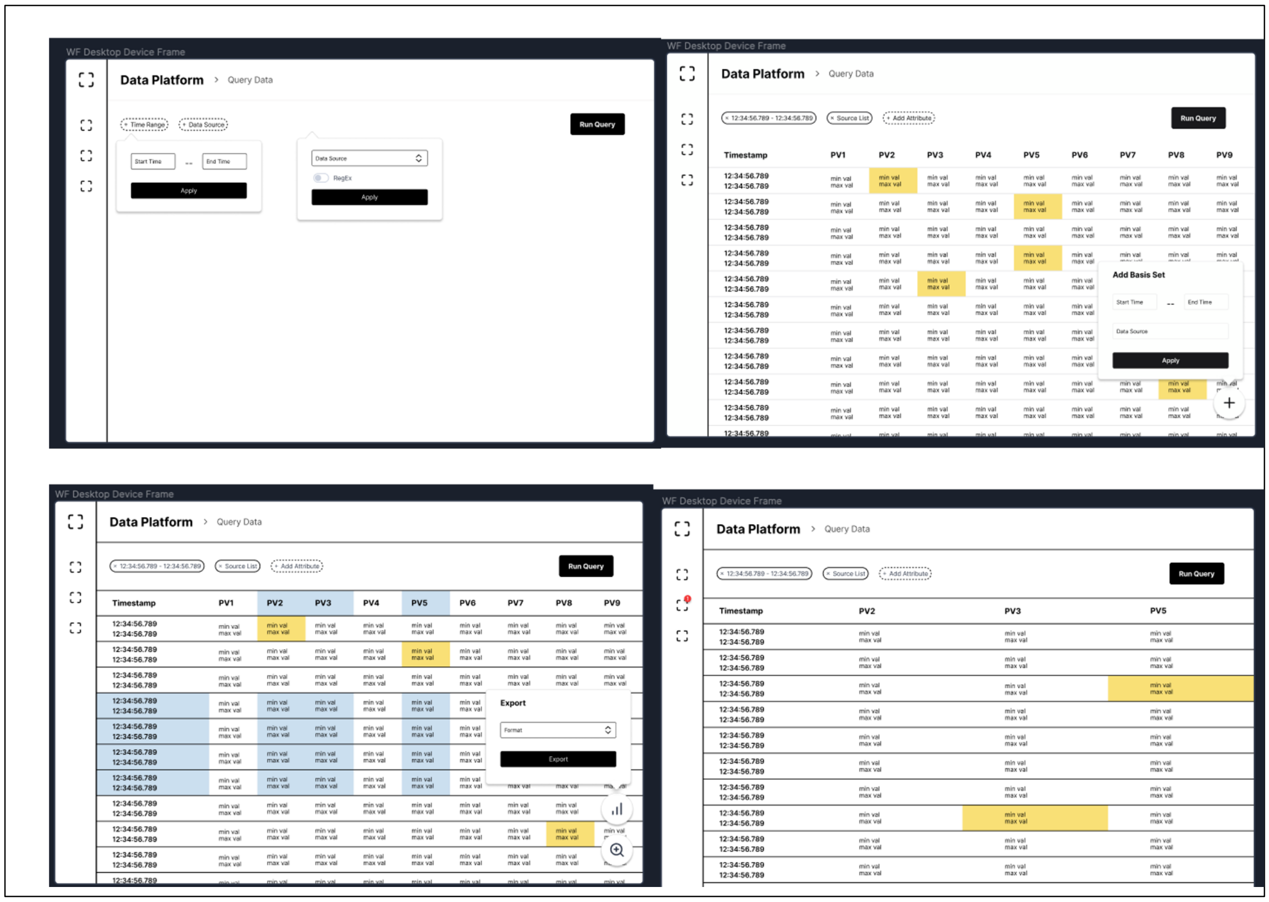
Stated in the Phase II narrative are the following requirements for the Web Application:

1. Data exporting of archived time-series data and metadata: Isolate and export archive data in common formats (CSV, Excel, NumPy, etc.). Data sets of interest are identified remotely then downloaded for local analysis and processing.
2. Provide common data science functions: Data visualization in the form of graphs and charts. The ability to analyze time-series data using standard data science techniques such as averages, fitting, correlations, supports, etc. Broad development in this area is unwarranted as specialized analysis is available through data exporting, common spreadsheet applications, and available data science tools and libraries.
3. Post-ingestion archive annotations (i.e., a viable subset): Features of interest are the ability to annotate archived data with user notes and comments, and to provide data associations within the archive.

In Quarter 2 Osprey DCS engaged a junior developer assigned to lead development for the Web Application. A significant portion of Quarter 2 was dedicated to training the new employee; this includes familiarization with the operation and architecture of the MLDP, the functions and feature requirements of the Web Application, and the tools and technologies needed for development.

The prototype Web Application had an elementary user interface, functional but not complete, nor intuitive. Thus, a dedicated effort was made on designing the new user interface with focus on comprehensive functionality, and with particular attention to the "look and feel." The objective is to provide a Graphical User Interface (GUI) having full functionality while simplifying user interaction using intuitive controls and options. Several screenshots of the new interface are shown in Figure 9. As seen in these examples, the user is presented an intuitive tabular view of the data archive. The interface allows the user to interact with data directly by highlighting and selecting various regions of interest for further processing. Users can drill down recursively within larger regions to isolate *data blocks* of interest. Multiple data blocks can be concurrently selected to build *data sets*, the targets for annotations and/or further query requests.

Additional properties for selected data (e.g., metadata) are also available using mouse over windowing and mouse right clicking.



WF Desktop Device Frame

Data Platform › Query Data

12:34:56.789 - 12:34:56.789 · Source List · Add Attribute · Run Query

Timestamp	PV1	PV2	PV3	PV4	PV5	PV6	PV7	PV8	PV9
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val	min val	min val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val			
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val			
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val			
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val			
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val
12:34:56.789									

WF Desktop Device Frame

Data Platform › Query Data

12:34:56.789 - 12:34:56.789 · Source List · Add Attribute · Run Query

Timestamp	PV1	PV2	PV3	PV4	PV5	PV6	PV7	PV8	PV9
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val			
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val			
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val			
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	max val	max val	
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	min val max val	
12:34:56.789									

WF Desktop Device Frame

Data Platform › Query Data

12:34:56.789 - 12:34:56.789 · Source List · Add Attribute · Run Query

Timestamp	PV2	PV3	PV5
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val
12:34:56.789 12:34:56.789	min val max val	min val max val	min val max val

Figure 9: Web Application user interface screenshots

For connection to the MLDP gRPC communication framework, the Web Application utilizes the "gRPC-Web" open library [3] [4]. The gRPC-Web framework is a JavaScript client library enabling browser-based applications to interact with gRPC-based services using an envoy proxy server [5]. The situation is illustrated in Figure 8. The necessity for this general setup (i.e., a gRPC proxy server) is because web browser applications utilize the HTTP transport protocol while gRPC is based upon HTTP2, thus some sort

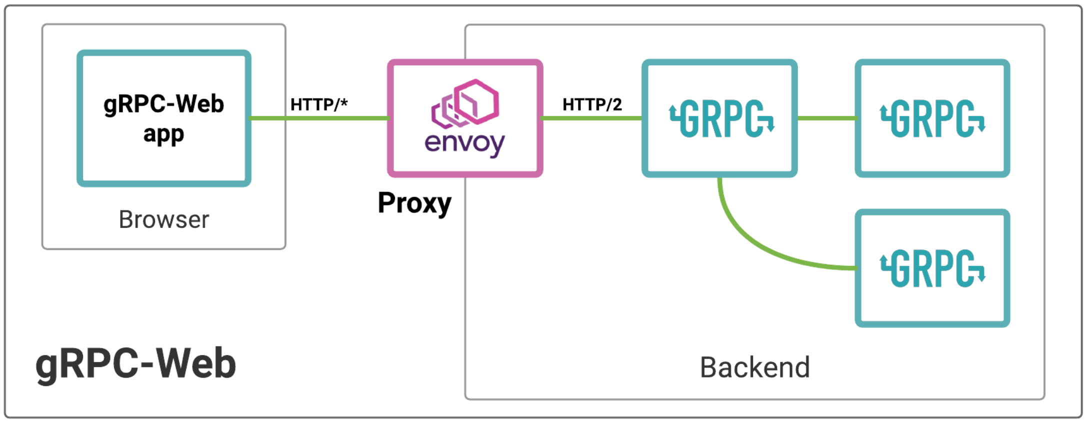


Figure 8: Web Application and gRPC Web framework (taken from ref. [4])

of "translation" is always required. The choice of Web-gRPC/envoy was made for its relative convenience, open-source distribution, and relatively mature technology (it is part of the gRPC framework under direct support).

The basic communications system for the Web Application has been designed and built, and the envoy services are now included with the MLDP installation. Successful interaction between the Web Application and the Query Service has been established. We are now building out the various features and actions seen in the examples of Figure 9, currently focusing on establishing all the data inspection and isolation features. The requirements described above will be addressed once this task is complete. Web Application development will continue in throughout the project duration as it is principally an independent task.

vi. Documentation

We wish to remark that a significant effort has been devoted to MLDP documentation in Quarter 2. The code repository hosting the Data Platform installation and deployment system also hosts the central MLDP documentation library. This library has been supplemented with relevant reports, documents, and presentations concerning the MDLP and its development. Additionally, many details concerning deployment, installation, operation, and implementation have been added to all MLDP repository documentation. This includes progress lists, task lists, bug reporting, issue tracking. The MLDP online documentation system is becoming increasingly formalized.

3. Training Opportunities and Professional Development

Significant professional development and training was afforded to M. Frauenheim in Quarter 2. Mr. Frauenheim has assumed lead role for Web Application development. To establish himself in this effort he engaged in substantial self-study along with interaction and training from project members C. McChesney and C.K. Allen. Mr. Frauenheim was able to familiarize himself with the MLDP, the necessary technologies, and multiple tools required for Web Application development and proper communications with the MLDP core services.

4. Dissemination of Project Results

Several members of the MLDP development team plan to attend the International Particle Accelerator Conference in Nashville, Tennessee (IPAC24). Abstracts concerning the current project results and status have been submitted. Additionally, Osprey DCS has provided notification and announcements of Data Platform development and the larger MLDP efforts to its current customer base and within the control system and accelerator community. Regarding the latter, there are now several projects and institutions anticipating the availability of a MLDP beta release, including Arizona State University (ASU), the National Aeronautics and Space Agency (NASA), and Stanford Linear Accelerator Complex (SLAC).

5. Activity Schedule for Next Quarter

Formal focus areas for Quarter 3 are the following: 1) Annotation Service, the 2) Web Application, 3) the Java language API library, and 4) Performance Investigations. These tasks are described in the following:

1) Annotation Service: The basic design and initial gRPC communications interface for the Annotation Service was completed in Quarter 2. The objective for Quarter 3 is to develop an operational system with basic functionality. We have initiated development of a basic service utilizing the design and principles described in Subsection I.2.ii above. The initial implementation will support "comment" annotations, the most basic annotation requirement but one that includes all aspects of the Annotation Service design described in Subsection I.2.ii. The intent is to complete this task within Quarter 3, with expectation that the "data association" feature could be at least initiated, if not completed. From the outset we recognized that the archive annotations feature of the MLDP would require significant development resources; our emphasis here is to create a robust, flexible design enabling future modifications and upgrades. That is, we prefer to "front load" the

development of the Annotation Service rather than rapidly build a system potentially requiring significant future redevelopment.

2) Web Application: Osprey DCS has hired a junior software developer who now leads the Web Application development effort. After initial orientation and training he is now successfully developing the web browser component of the application. The preliminary design of the Query Service Graphical User Interface (GUI) has been completed. The Quarter 3 efforts will focus on building out the user interactions with the Query Service as available in the GUI interface. (This necessarily involves establishing robust browser communications with the core services gRPC framework.) Quarter 3 development also anticipates fitting the Query Service GUI with some basic data science and/or visualization features, such as graphing and basic statistical functions for time-series data.

3) Java Language API Library: Due to the new MLDP design and gRPC communications framework the previous programming language client API libraries are no longer viable. Rather than attempting to port the previous Java client API library to the new gRPC framework and core services functionality, we are rebuilding the Java client library (i.e., by leveraging off the previous work). The gRPC connection mechanism and library configuration mechanism both were built in Quarter 2, along with a functioning Query Service interface (although incomplete). The Quarter 3 focus is development of the Ingestion Service interface within the library. Development of the Java library interface for the Annotations Service will not proceed until the service is mature, likely at the end of Quarter 3 or start of Quarter 4.

4) Performance Investigation: There are additional performance possibilities that we also wish to investigate in Quarter 3. Currently all time-series data is stored in MongoDB native format. We intend to examine the direct storage of compressed time-series data, utilizing MongoDB as the primary bookkeeping mechanism. This idea is attractive since time-series data is always serialized by Protocol Buffers for network transport by gRPC. We have found this serialization mechanism to be extremely efficient while also providing a conspicuous 10% data compression. We expect that storing the serialized data directly may yield a significant performance benefit, as well as a significant savings in storage space. As a further performance benefit the stored data will not require re-serialization by the Query Service for the raw-data transport option. Additionally, the storage of serialized data offers a cheap solution to the archiving of structured heterogeneous data. The cost of maintaining bookkeeping information concerning all serialized data sets will likely be the determining factor in this approach.

In addition to the above focus areas, once the Ingestion Service interface is available with the Java client API library, there are additional activities we wish to pursue starting in Quarter 3.

5) We intend to port data simulator tool previously developed in Phase I for operation with the new Java API library. The data simulator provides a rich set of ingestion scenarios for testing and benchmarking the Ingestion Service (i.e., beyond that of the Administration Library components developed in Quarter 2).

6) Ospreys DCS is in current contract with NASA concerning high-speed data acquisition. We intend to test archiving of the acquired data within the MLDP data archive; this task requires specialized data formatting possible with the Java API library Ingestion Service interface.

There are other activities that may be targeted if the opportunity presents itself within Quarter 3. Thus, we include the further comments regarding our development approach and intent.

- The Query Service still has additional features to build out, which will proceed as necessary. It is likely that the development efforts for the Annotation Service may yield serendipitous results which can be applied to the Query Service implementation. Specifically, the notation that the "data set" described in Subsection I.2.ii is essentially a query request may produce a more streamlined approach to query operations: this idea was presented in Section I.2.ii when describing the Annotation Service.

- Programming language support for MLDP communications greatly simplifies client interaction and is particularly desirable for backend users of the Query Service and Annotations Service, specifically supporting data-science and machine-learning application development by data scientists and engineers. A client API library for the Python language is planned, one based upon the Pandas library familiar to most data scientists. Direct support for the DeepLearning4Java library is also planned. Development here is anticipated in the first year, as opportunity permits.
- Although development for the Web Application is essentially independent and can proceed at any time during the project (i.e., once the Query Service and Annotation Service are operational), we elected to begin development early. The Web Application offers tremendous potential in the dissemination of project activities and results. Being deployed within a standard web browser and being supported by an independent host server, the Web Application requires only commonly available resources to demonstrate many of the commercial features and standard use cases for the MLDP.

II. PRODUCTS

1. Publications

No formal publications have yet been produced. Journal publication(s) are anticipated as a final product of the project once the technologies required for fast heterogeneous data management are mature and/or the MLDP has been used successfully in real-world application. Publicly available conference proceedings concerning current development efforts, products, and status of the MLDP are scheduled for the IPAC24 conference in Nashville, Tennessee.

Note that there is now an increasing body of publicly available documentation on the MLDP in the form of technical reports and presentations. These documents are available within the MLDP installation repository [6]. Additionally, there is substantive online documentation available within the development and distribution repositories for the MLDP. Additionally, an internal document detailing C++ gRPC implementations has been produced [7]; it contains necessary information for future development of the Advanced Data Science systems to commence in Year 2 (also available within the installation repository).

2. Intellectual Property

Osprey DCS claims no intellectual property rights over the current SBIR project results. All products are to be open source and freely available to the community.

3. Technologies and Techniques

A main component of the MLDP core, the Query Service, has been developed in Quarter 2 and is available as a development release within the MDLP installation. This is in addition to the Ingestion Service component developed in Quarter 1. The Query Service has most basic features operational and has been tested extensively. Additionally, it performs as expected in concert with the Ingestion Service, as extensive data throughput testing has been performed. The Ingestion Service and the Query Service sub-systems are technologies and techniques for the fast storage and retrieval of heterogenous, time-series data and metadata necessary for data science applications within a large particle accelerator facility or other large experimental facility within the Department of Energy complex.

Another significant technology/technique currently under development is the ability to supplement archived data with user notes, relationships, and calculations, or more generally "user annotations." This technology provides a "value-added" capability to the data archive created from user interaction.

Note that since Osprey DCS claims no property rights over these technologies and techniques, they are to be shared with the accelerator and experimental physics community through open-source availability of public code repositories.

4. Other Products

The MLDP installation and deployment system has also been greatly improved, development releases of the MLDP are available on GitHub for inspection[2] [6]. The deployment system provides direct installation via an installer archive that is downloaded using a standard web browser. There are instructions for configuring the installation to specific platform on the installation website[3]. The Ingestion Service, the Query Service, and companion benchmarking and validation utilities are all included in the current development release installation. The installer has full functionality and can locally deploy current development releases of the Data Platform (i.e., MDLP core services) for testing and external inspection.

III. PARTICIPANTS AND COLLABORATING ORGANIZATIONS

All design and development activities within the first two quarters have been internal to Osprey DCS. External collaboration is premature until the MLDP communications framework and APIs are stable and well established. This situation will likely remain throughout the first year.

External organizations have expressed interest in utilizing MLDP components as they become available. As particular instances mentioned previously, ASU and SLAC have expressed interest in testing initial deployments of the MLDP. Additionally, once the Java client API library Ingestion Service interface is available, Osprey DCS intends to integrate the MDLP archiving system into a fast-data acquisition system currently being developed for NASA.

Early in the project the Sierra Peaks company contracted with Osprey DCS for control systems design and development support concerning their efforts on a current DOE project (SCORPIUS); the contract included specific investigations of the MLDP core services as a system component. Details are provided below in Section III.2.

1. Participants

1) Name: Bob Dalesio
2) Project Role: Principal Investigator
3) Effort (months): 1
4) Contributions: Project management, high-level MLDP architecture.
5) Foreign Collaboration: No.
6) Foreign Travel: No

1) Name: Craig McChesney
2) Project Role: Senior Developer
3) Effort (months): 3
4) Contributions: Data Platform redesign, Ingestion Service development, deployment systems.
5) Foreign Collaboration: No
6) Foreign Travel: No

[2] Access to the MLDP installation repository is still currently private but scheduled for public release once we are confident in the stability of the gRPC interface. Interested parties may contact Osprey DCS for access.

[3] The local platform must have Java version 16 (or greater) and MongoDB Community Edition 6.0 (or greater) installed. There are instructions for doing so on the installation website.

1) Name: Christopher K. Allen
2) Project Role: Senior Developer
3) Effort (months): 3
4) Contributions: Data Platform design studies, datastream processing investigations.
5) Foreign Collaboration: No
6) Foreign Travel: No

1) Name: Michael Davidsaver
2) Project Role: Senior Developer
3) Effort (months): 1
4) Contributions: Data Platform design.
5) Foreign Collaboration: No
6) Foreign Travel: No

1) Name: Mitch Frauenheim
2) Project Role: Junior Developer
3) Effort (months): 2
4) Contributions: Web Application primary developer.
5) Foreign Collaboration: No
6) Foreign Travel: No

2. Partners

Sierra Peaks was briefly involved with investigations of the Datastore prototype before the Phase II project was initiated. Sierra Peaks company contracted with Osprey DCS for control systems design and development support concerning their efforts on a current DOE project (SCORPIUS). Their staff discussed development and system integration with aspects of the MLDP.

1) Organization Name: Sierra Peaks
2) Location: Albuquerque, New Mexico
3) Contribution: Consultation
4) Financial Support: $30,000 - contracted for control system support (included use of Data Platform)
5) In-kind Support: None
6) Facilities: None
7) Collaborative Research: None
8) Personnel Exchanges: None

3. Other Collaborators

No external organizations have yet participated in any formal development of MLDP systems. However, Osprey DCS is currently pursuing collaborations with Arizona State University in support of the CXLS and CXFEL accelerator systems. The intent is to deploy the MLDP system to collect and analyze machine and experimental data.

IV. IMPACT

As the current SBIR Phase II project is in its initial stage, direct impact on particle accelerator systems has yet to be determined. However, there is a growing interest in the project as we disseminate our current activities within the community.

It is anticipated that the technologies developed for rapid storage and retrieval of heterogeneous data will have a significant influence in the areas of data systems and data transport. The fact that we already achieved the performance milestone for the Ingestion Service is significant.

1. Impact on Principle Discipline

Since the MLDP is not yet deployed in the field, no explicit impact studies are available. However, it is anticipated that the availability of the MLDP will significantly enhance the applicability of machine learning and general data science techniques to the operation and optimization of accelerator systems. Additionally, the MLDP can support analysis of experimental data collected at accelerator facilities and other large experimental physics facilities within the Department of Energy complex and industry.

2. Impact on Other Disciplines

The MLDP is a tool for fast data acquisition and archiving that supports general machine learning and data science techniques applied to correlated, heterogeneous, time-series data. Although the data acquisition and aggregation systems are tailored to facilities using the EPICS control system, the data ingestion and archiving systems are independent. Thus, any facility conforming to the Ingestion Service communications protocol can archive data within the MLDP and, thus, utilize all back-end data science capabilities, including the Web Application. Due to this generality, there is expectation of broader application beyond particle accelerator systems.

3. Impact on Development of Human Resources

The MLDP may be considered a productivity tool for personnel in the data sciences.

4. Impact on Physical, Institutional, and Information Resources

The Machine Learning Data Platform is a tool for acquisition, collection, and dissemination of information pertaining to facility operations and experimental data. Data availability is tailored to machine learning and data science personnel and applications. Additionally, the companion Web Application provides universal remote access and interaction with all archived data. The intent is to make facility operations data and experimental data as widely available and accessible as possible.

5. Impact on Technology Transfer

No techniques, technologies, tools, or resources developed in this project are considered proprietary or business sensitive. Consequently, all products are open source and may be freely transferred throughout the community.

6. Impact on Society Beyond Science and Technology

Unknown.

7. Foreign Spending

No project expenditures have been made in or to foreign countries. None are expected in the future.

V. CHANGES AND PROBLEMS

No significant modifications to the project approach or schedule have been made. The core services of the original MLDP prototype have been redesigned and rebuilt, however, this was a possibility that was anticipated and addressed in the SBIR Phase II Project Narrative. However, the details of the final design differ from preliminary designs offered in the original narrative, as they are the result of exhaustive design and performance studies conducted at Phase II inception.

1. Changes in Approach

Some Advanced Data Science investigations were performed early, as they could be conducted in parallel with the initial design studies. The MLDP installation and deployment system was developed early as it assisted significantly in team development. The Web Application development has also started early as the opportunity was available.

2. Actual and/or Anticipated Problems

As described above, the core services of the Machine Learning Data Platform (MLDP) required a complete redesign and rebuild to meet project performance requirements. The likelihood of such an action was recognized in the Phase I Final Report [2] and included in the original Phase II Project Narrative. The Phase II schedule included estimates of time and effort if this action was necessary.

Development for the Annotations Service has been initiated according to the original schedule. However, due to its sophistication we have decided to "front-load" the process as it is a sophisticated system providing a critical feature of the MLDP. Thus, development in this area may exceed original predictions.

3. Changes Requiring Significant Impact on Expenditure

No changes in project expenditures have been incurred.

4. Significant Changes in Use of Human Subject, Animals, and/or Biohazards

The project uses no animals or hazardous materials. Other than feedback and critiques from MLDP users, there are no human subjects or experimentation.

5. Changes in Primary Performance Site Location

There are currently no changes in site locations.

6. Carryover Amount

There are no project carryover amounts from this quarter.

VI. DEMOGRAPHIC INFORMATION

Listed below are the email addresses of all project participants. All participants are available for contact via email concerning any demographic information requests.

Bob Dalesio: bdalesio@ospreydcs.com

Craig McChesney: cmcchesney@ospreydcs.com

Christopher K. Allen: allenck@ospreydcs.com

Michael Davidsaver: mdavidsaver@ospreydcs.com

Mitchell Frauenheim: mfrauenheim@ospreydcs.com

VII. SPECIAL REPORTING REQUIREMENTS

There are currently no special reporting requirements for this project.

REFERENCES

[1] Osprey DCS, "Research Performance Progress Report: Quarter 1, DE-SC0022583," Ocean City, Maryland, 2023.

[2] C. Allen et. al., "Machine Learning Data Platform: Phase I Research and Development Report," Osprey Distrubuted Control Systems, LLC, Ocean City, Maryland, 2023.

[3] M. Diamond, "grpc-web," GitHub, 2024. [Online]. Available: https://github.com/grpc/grpc-web.

[4] L. Perkins, "Envoy and gRPC-Web: a fresh new alternative to REST," Medium, 2018. [Online]. Available: https://blog.envoyproxy.io/envoy-and-grpc-web-a-fresh-new-alternative-to-rest-6504ce7eb880.

[5] Envoy Project, "envoy," Cloud Native, 2024. [Online]. Available: https://www.envoyproxy.io/.

[6] Osprey DCS, "Data Platform," Osprey Distributed Control System, LLC, 2024. [Online]. Available: https://github.com/osprey-dcs/data-platform.

[7] C. K. Allen, "Evaluation of C++ gRPC," Osprey DCS, TM-2023-003, Ocean City, Maryland, 2023.