ORNL/TM-2022/2717

INTERSECT Architecture Specification: System-of-Systems Architecture (Version 0.9)

Olga A. Kuchar, Swen Boehm, Thomas Naughton, Suhas Somnath, Ben Mintz, Jack Lange, Scott Atchley, Rohit Srivastava, Patrick Widener

September 30, 2023

Approved for public release.
Distribution is unlimited.

OAK RIDGE
National Laboratory

ORNL/TM-2022/2717

Laboratory Directed Research and Development Program
Self-Driven Experiments for Science/Interconnected Science Ecosystem (INTERSECT) Initiative

INTERSECT Architecture Specification: System-of-Systems Architecture (Version 0.9)

Olga A. Kuchar, Swen Boehm, Thomas Naughton, Suhas Somnath, Ben Mintz, Jack Lange, Scott Atchley, Rohit Srivastava, Patrick Widener

September 30, 2023

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

LIST OF FIGURES . . . v
LIST OF TABLES . . . ix
LIST OF EXAMPLES . . . xi
Acronyms and Abbreviations . . . xiii
INTERSECT Terminology . . . xv
Acknowledgements . . . xvii
Abstract . . . xix
Revision Record . . . xxi
1. Introduction . . . 1
  1.1 INTRODUCTION . . . 1
  1.2 PURPOSE OF THIS DOCUMENT . . . 3
  1.3 STAKEHOLDER REPRESENTATION . . . 3
  1.4 DOCUMENT SCOPE . . . 5
  1.5 DOCUMENT OVERVIEW . . . 5
2. Logical View . . . 7
  2.1 INTRODUCTION . . . 7
  2.2 ARCHITECTURE OVERVIEW . . . 7
  2.3 ARCHITECTURE CONCEPTS . . . 8
  2.4 OBJECT MODEL . . . 12
  2.5 SYSTEM DESCRIPTIONS . . . 19
  2.6 SYSTEM OPTIONS . . . 31
3. Operational View . . . 33
  3.1 INTRODUCTION . . . 33
  3.2 HIGH-LEVEL OPERATIONAL DIAGRAM . . . 34
  3.3 OPERATIONAL ACTIVITIES . . . 37
4. User View . . . 47
  4.1 INTRODUCTION . . . 47
5. Data View . . . 51
  5.1 INTRODUCTION . . . 51
  5.2 ENTITY RELATIONSHIP DATA MODEL . . . 51
  5.3 INTERSECT DATA MESSAGING SCHEMA . . . 53
  5.4 EXAMPLE MESSAGING SEQUENCE DIAGRAMS . . . 66
6. Physical View . . . 71
  6.1 INTRODUCTION . . . 71
7. Standards View . . . 75
References . . . 76
Appendices . . . 77

A. USER VIEW WIRE DIAGRAMS . . . 78
A.1 USER PERSON TYPE AND ASSOCIATED VIEWS . . . 78
A.2 OWNER . . . 128
A.3 OPERATOR / MAINTAINER (OM) . . . 140
A.4 ADMINISTRATOR . . . 149
B. INTERSECT MESSAGE SCHEMA . . . 151

LIST OF FIGURES

1.1 High-level illustration of INTERSECT tools with a System of System Architecture. 2
2.1 Differentiation of logical and infrastructure systems. 8
2.2 Logical Systems in the INTERSECT architecture 9
2.3 An abstract example of the relationship of task, command and action 13
2.4 A concrete example of the relationship of task, command and action 13
2.5 Representation of the capability object 14
2.6 Representation of the Agent Object 15
2.7 Task states 16
2.8 Relationship of different systems to provide a container as a compute capability 19
2.9 Relationship of different systems to provide a batch job as a compute capability 20
2.10 General capabilities of the Launching Service. 21
2.11 Instrument adapter capabilities. 22
2.12 Example json document describing an instrument 23
2.13 Overview of Data Management 24
2.14 File System 25
2.15 Data Managment 26
2.16 Container Orchestration 27
2.17 Tasks 27
3.1 High-level diagram to assist operational view of system; showing different facility domains that will be managed by different Operators and Administrators depending on the resource and Facility. 34
3.2 High-Level View of Systems & Sub-systems High-Level View of Systems & Sub-systems with a brokered messaging paradigm for communication. 35
3.3 Diagram showing hierarchy of Actions, Commands and Command-Sets. 36
3.4 Hierarchical naming structure for System of System addressing. 37
3.5 Illustration of messages involved in service registration to obtain a universally unique identifier (UUID). 38
3.6 Sequence diagrams for illustration of Compute Service Monitoring using two possible approaches. Note, Blue box is INTERSECT Service, Green box/region is resource/opaque system. 41
3.7 INTERSECT Service Monitor illustrating sequence diagram for interfacing with centralized Monitoring Service that stockpiles data about the various INTERSECT Services (e.g., Compute, Storage, etc.). Note, Blue box is INTERSECT Service, Green/Yellow/Grey boxes/regions are resources/opaque systems. 42

3.8 Illustration of "Push" model form of Computer Service Monitoring where Actor is updated asynchronously about monitoring status updates, and the ComputerSvcMon polls resources periodically about status and publishes to Message Broker that disseminates the messages to subscribers. 43
3.9 Sequence diagram for experiment planning workflow that shows an ApprovalRequest/ApprovalRequestStatus message exchange. 46
5.1 Entity-Relationship Diagram for INTERSECT. 54
5.2 Top level of the INTERSECT XML schema. 55
5.3 Sequence diagram for INTERSECT component registration. 67
5.4 Sequence diagram for user login to INTERSECT. 68
5.5 Sequence diagram for creating and scheduling a campaign on INTERSECT. 69
5.6 Running a campaign on INTERSECT. 70
A.1 A representation of the information that a new or potential user of INTERSECT would be required to register for an INTERSECT account.. 79
A.2 A representation of the home-screen that would be displayed once a User logs into the INTERSECT web interface.. 80
A.3 A representation of a registered User's profile in INTERSECT with options to edit their profile. 83
A.4 A representation of a User's project view.. 86
A.5 A high-level view representing past, present, and future scientific Campaigns that this user has been part of. 88
A.6 Detailed view of a single scientific Campaign as viewed by a participant (composer, Owner or Operator of a requested Resource, Administrator) of the Campaign. This view focuses on the general information and detailed status of the primary Resources requested in this Campaign (once the Campaign is submitted) 92
A.7 Detailed view of a single scientific Campaign as viewed by a participant (composer, Owner or Operator of a requested Resource, Administrator) of the Campaign. This view focuses on the workflow configuration for the Campaign. 93
A.8 Buttons are shown for different campaign types. 95
A.9 When a campaign is selected, a basic template is shown. 96
A.10 A list of the most common resources is shown. 98
A.11 Filters can be applied to specify a search. 98
A.12 Detailed view of a physical resource. 103
A.13 Detailed view of a logical resource.. 104
A.14 Prompt to request access to a resource for a campaign. 105
A.15 Notification panel is specific to each user. 107
A.16 Users can choose which notifications to get and how to get them. 109
A.17 These fields are prompted regardless of chosen template. 113
A.18 User must configure run-time behavior before starting campaign. 114
A.19 User authenticates resource for the campaign. 116
A.20 A dashboard of recent compaigns is shown at the end of the campaign. 118
A.21 Form to add resource to INTERSECT. 126
A.22 Capability that resource is capable of supporting. 127
A.23 Process perspective of logging into INTERSECT and building/executing a workflow. *This figure will be updated in a future version of this document.* 127

A.24 Owner's view of resource notificaitons regarding access. 129
A.25 This management is done in project. 131
A.26 Three options are given regarding access. 131
A.27 Owners can see who is contributing to what resource. 134
A.28 A list of resources separated by Type, Category, and Provider. 135
A.29 Options are given to temporarily or permanently remove the resource. 139
A.30 Operator/Maintainer notification panel. 141
A.31 Operator/Maintainer monitors resource. 143
A.32 This take down takes places in the detailed view. 146
A.33 Alternates can be suggested if the three choice do not suffice. 148

LIST OF TABLES

1.1 Stakeholder roles and the views within this document. 4
4.1 Person Type roles and descriptions. 48
5.1 Names and descriptions of INTERSECT architecture data entities 52

List of Examples

Example 2.1 10
Example 2.2 11
Example 2.3 11
Example 2.4 11
Example 2.5 11
Example 2.6 12
Example 2.7 12
Example 2.8 14
Example 2.9 15
Example 2.10 16
Example 2.11 17
Example 2.12 18
Example 2.13 19
Example 2.14 19

ACRONYMS AND ABBREVIATIONS

AI artificial intelligence. 1, 3
API application programming interface. xv, 11, 78
blob binary large object. 12
CADES ORNL Compute and Data Environment for Science. 36
CLI Command Line Interface. 78
CMS Campaign Management System. 25
CNMS Center for Nanophase Materials Sciences. 34, 36
CPU central processing unit. 16
DMS Data Management System. 23, 24
DoDAF Department of Defense Architecture Framework. 3, 7, 47
DOE U. S. Department of Energy. 1
DTN Data Transfer Node. 36
E-R entity-relationship. xxi
fqdn fully qualified domain name. 11
GUI graphical user interface. 49, 78, 110
HMI human-machine interface. 1
IAT INTERSECT Architecture Team. xvi
IEEE Insitute of Electrical and Electronics Engineers. 3
IMS Infrastructure Management System. 19
INTERSECT Self-driven Experiments for Science / Interconnected Science Ecosystem. 1–3, 7–12, 14, 15, 19, 20, 22–25, 30, 34, 36, 37, 39, 40, 47, 48, 78, 80, 97
LSI lead system integrator. 4
OLCF Oak Ridge Leadership Computing Facility. 34, 36
ORNL Oak Ridge National Laboratory. 1, 34
SoS system of systems. 1–3, 37
UI User Interface. 11
UMS User Management System. 25
URI Uniform Resource Identifier. 10–12, 53
UUID universally unique identifier. 12, 37

INTERSECT TERMINOLOGY

Activity	Work, not specific to a single organization, system or individual that transforms inputs (Resources) into outputs (Resources) or changes their state. [DODAF,IAT]
Administrator	Role of entity that maintains one or more systems; complete view of "their" system (their jurisdiction/domain/realm/area); Limited to a given jurisdiction (i.e., their administrative domain). [IAT]
Administrative Domain	Jurisdiction of control. [IAT]
Agent	Something that produces or is capable of producing an effect: an active or efficient cause. [MW]
Campaign	A scientific endeavor that may consist of one or more Experiments that may take place sequentially or in parallel to answer a broader overarching scientific question. [IAT]
Capability	The ability to achieve a desired effect under specified [performance] standards and conditions through combinations of ways and means [activities and resources] to perform a set of activities. [DODAF]
Experiment	An indivisible component of a scientific endeavor that typically involves measurements, computation, and/or data analysis. An experiment is performed with a unique set of conditions and/or parameters. So long as the parameters are feasible, every Experiment will have a clear start and a predicable end. Insights of an experiment are often narrow and may not answer broader scientific questions. [IAT]
Facility	A real property entity consisting of underlying land and one or more of the following: a building, a structure (including linear structures), a utility system, or pavement. [DODAF]
Institution	See Organization.
I-System	Abbreviation for an "INTERSECT System". [IAT]
Jurisdiction	Realm of authority; Administrative domain. [IAT]
Maintainer	See Operator.
Microservice	A service that is designed according to the Microservices Architecture methodology and that implements a Microservice Contract for its clients. [IAT]
Microservice Architecture	A design methodology for structuring a distributed Application as a networked collection of loosely-coupled services that are independently developed, maintained, and operated. [IAT]
Microservice Function	A specific Microservice functionality with clearly defined input and output data. The function may have associated service state pre-conditions and/or post-conditions. [IAT]
Microservice Contract	The complete set of Microservice Functions provided by a Microservice to its clients, typically scoped using domain-driven design and defined using an application programming interface (API). [IAT]
Operator	Role of entity that maintains one or more resources; different view of system (i.e., in contrast to User). [IAT]

Organization	A specific real-world assemblage of people and other resources organized for an on-going purpose. [DODAF]
Owner	Role of entity fiscally responsible for a resource; Vested interest; Possibly approver for a resource. [IAT]
Performer	Any entity - human, automated, or any aggregation of human and/or automated - that performs an activity and provides a capability. [DODAF]
PersonType	A category of persons defined by the role or roles they share that are relevant to an architecture. [DODAF]
Provider	Role of entity that manufactures a resource (intention is distinction between provider/owner), e.g., SecDevOps is provider of SDK. Provider creates the Service that the Operator maintains. Provider creates the Instrument that the Operator maintains. [IAT]
Resource	Data, Information, Performers, Materiel, or Personnel Types that are produced or consumed. [DODAF]
Role	Performs a specific function; implies access rights for resources; checks for adherence to resource/jurisdiction/facility polices. [IAT]
Service	A mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. The mechanism is a Performer. The "capabilities" accessed are Resources – Information, Data, Materiel, Performers, and Geo-political Extents. [DODAF]
Sub-system^1	A self-contained system within a larger system that is capable of both independent operation as well as coordinated interaction with other systems. [IAT]
Sub-system^2	(Alt. definition) A self-contained system within a larger integrated System. [IAT]
System (or I-System)	A logical entity with operational and managerial independence that provides utility to the overall System of Systems. A System may utilize one or more physical resources and may be geographically distributed. Systems communicate with each other with the INTERSECT protocol for control purposes. A System provides this utility via one or more Services. A System may have Sub-systems. [IAT]
Task	A function to be performed; an objective; a usually assigned piece of work often to be finished within a certain time. [IAT]
User	Role for entity using the system (not responsible for administration). [IAT]
Workflow	Activities that are performed according to a recipe (i.e. sequential, in a DAG) or script. Static or Dynamic Workflow. [IAT]

Sources

[DoDAF]	DoD Architecture Framework v2.02 Glossary, https://dodcio.defense.gov/Library/DoD-Architecture-Framework/dodaf20_glossary
[IAT]	INTERSECT Architecture Team
[MW]	Merriam-Webster

ACKNOWLEDGEMENTS

This report describes products of research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory (ORNL). ORNL is managed by UT-Battelle LLC on behalf of the U. S. Department of Energy.

ABSTRACT

Oak Ridge National Laboratory (ORNL)'s Self-driven Experiments for Science / Interconnected Science Ecosystem (INTERSECT) architecture project, titled "An Open Federated Architecture for the Laboratory of the Future", creates an open federated hardware/software architecture for the laboratory of the future using a novel system of systems (SoS) and microservice architecture approach, connecting scientific instruments, robot-controlled laboratories and edge/center computing/data resources to enable autonomous experiments, "self-driving" laboratories, smart manufacturing, and artificial intelligence (AI)-driven design, discovery and evaluation.

The architecture project is divided into three focus areas: design patterns; SoS architecture; and microservices architecture. The design patterns area focuses on describing science use cases as design patterns that identify and abstract the involved hardware/software components and their interactions in terms of control, work and data flow. The SoS architecture area focuses on an open architecture specification for the federated ecosystem that clarifies terms, architectural elements, the interactions between them and compliance. The microservices architecture describes blueprints for loosely coupled microservices, standardized interfaces, and multi-programming language support.

This document is the SoS Architecture specification only, and captures the system of systems architecture design for the INTERSECT Initiative and its components. It is intended to provide a deep analysis and specification of how the INTERSECT platform will be designed, and to link the scientific needs identified across disciplines with the technical needs involved in the support, development, and evolution of a science ecosystem.

This working document reflects current discussions and design activity among the authors. The authors have worked to eliminate significant inconsistencies in publicly available versions. Comments from readers are welcome as we continue to evolve this document and the INTERSECT SoS design it describes.

REVISION RECORD

Table of Revisions

Version	Date	Description
0.9	09/30/2023	Public release with the following changes: • Incorporate feedback from external reviewers • Edits throughout for style and consistency
0.8	06/30/2023	Early draft release with the following changes: • Updates to Logical View to capture key concepts • Editorial updates for style and formatting
0.7	04/04/2023	Internal draft release with the following changes: • Edits throughout for style and consistency • Introduction: remove redundant document revisions list • User View: migrated wireframes to appendix • Logical View: add capability concept diagrams, other edits • Operational View: no content changes • Data View: add table describing entities in ER model • Physical View: no content changes • Standards View: added stub section to document
0.6	01/10/2023	Internal draft release with the following changes: • Edits throughout for style and consistency • User View: Migrated content from Data View which properly belongs here • Data View: Add INTERSECT message descriptions automatically generated from XML Schema. Revise entity-relationship (E-R) diagram based on changes in User View (not yet finalized). • Operational View: No content changes • Logical View: clarification of logical and infrastructure systems; added list of objects; improved document structure; general edits throughout • Physical View: No content changes
0.5	09/28/2022	Initial public release. Changes include: • Migrate document to LaTeX • User View: New wireframes were added starting from 4.2.9 onwards • Data View: Update entity-relationship diagram and messaging schema definitions • Operational View: add Command/Actions to high-level descriptions (3.2); add “Approval Processes” activity (3.3); formatting and text cleanups • Logical View: Update system concepts (2.3) and formatting improvements • Physical View: Added description of process for developing site specific physical views

Continued on next page

Table of Revisions – continued from previous page

Version	Date	Description
0.4	06/30/2022	Internal draft release with the following changes: • User View: updated wireframes and user scenarios. • Data View: updated INTERSECT schema and provided detailed descriptions for each component. • Operational View: reformatted section to have "activities"; updated text for Service Monitoring activity; added details for Service Registry activity; added details for Health Monitoring activity. • Logical View: added Section *[section removed]*; Edits sections: 2.1, 3.1-3.3, A.1-A.3, 5.3, 7, Appendix B, and updated wording throughout. • Physical View: major updates throughout entire section, with new network layouts for different facilities. Changes Sections 2.1, 3.1-3.3, A.1-A.3, 5.3, 7 and Appendix B.
0.3	04/04/2022	Internal draft release with the following changes: • Overall document formatting improvements, added Glossary, updated table of acronyms, added high-level summary text, and SoS illustration. • User View: added details regarding each user action; added wireframes of INTERSECT dashboard for specific user actions. • Data View: improvements to use case descriptions and entity relationship (ER) diagrams. • Operational View: added general system/subsystem and sequence diagrams; added details on service monitoring; formatting fixes. • Logical View: added new terms and constructs for system/subsystem/ adapter; added diagrams for modelling of logical elements of architecture. All chapters have been updated.
0.2	01/05/2022	Draft document released – internal
0.1	12/17/2021	Original document

1 INTRODUCTION

This section provides a high-level overview: the purpose of this document, the architectural views, the stakeholders addressed, what this document includes and doesn't include, and a brief summary of each major section.

1.1 INTRODUCTION

The U. S. Department of Energy (DOE)'s artificial intelligence (AI) for Science report [4] outlines the need for intelligent systems, instruments, and facilities to enable science breakthroughs with autonomous experiments, "self-driving" laboratories, smart manufacturing, and AI-driven design, discovery and evaluation. The DOE's Computational Facilities Research Workshop report [1] identifies intelligent systems/facilities as a challenge with enabling automation and eliminating human-in-the-loop needs as a cross-cutting theme.

Autonomous experiments, "self-driving" laboratories and smart manufacturing employ machine-in-the-loop intelligence for decision-making. Human-in-the-loop needs are reduced by an autonomous online control that collects experiment data, analyzes it, and takes appropriate operational actions to steer an ongoing or plan a next experiment. It may be assisted by an AI that is trained online and/or offline with archived data and/or with synthetic data created by a digital twin. Analysis and decision making may also rely on rule-based approaches, causal models, and advanced statistical methods. Human interaction for experiment planning, observation and steering is performed through a human-machine interface (HMI).

A federated hardware/software architecture for connecting instruments with edge and center computing resources is needed that autonomously collects, transfers, stores, processes, curates, and archives scientific data in common formats. It must be able to communicate with scientific instruments and computing and data resources for orchestration and control across administrative domains, and with humans for critical decisions and feedback. Standardized communication and programming interfaces are needed that leverage community and custom software for scientific instruments, automation, workflows and data transfer. Pluggability is required to permit quickly adaptable and deployable solutions, reuse of partial solutions for different use cases, and the use of digital twins, such as a virtual instrument, robot, or experiment. This federated hardware/ software architecture needs to be an open standard to enable adoption by DOE's science facilities.

Oak Ridge National Laboratory (ORNL)'s Self-driven Experiments for Science / Interconnected Science Ecosystem (INTERSECT) architecture project, titled "An Open Federated Architecture for the Laboratory of the Future", creates an open federated hardware/software architecture for the laboratory of the future using a novel system of systems (SoS) and microservice architecture approach, connecting scientific instruments, robot-controlled laboratories, and edge/center computing/data resources to enable autonomous experiments, "self-driving" laboratories, smart manufacturing, and AI-driven design, discovery, and evaluation.

This Architecture project describes science use cases as design patterns that identify and abstract the involved hardware/software components and their interactions in terms of control, work and data flow. It creates a SoS architecture of the federated hardware/software ecosystem that clarifies terms, architectural elements, the interactions between them, and compliance. It further designs a federated microservice architecture, mapping science use case design patterns to the SoS architecture with loosely coupled microservices, standardized interfaces, and multi programming language support. The primary deliverable of this project is an INTERSECT Open Architecture Specification, containing the science use case design pattern catalog, the federated SoS architecture specification, and the microservice architecture specification.

System of Systems Architecture

This document represents the federated SoS description of the INTERSECT Open Architecture Specification. The SoS architecture decomposes the federated hardware/software ecosystem into smaller and less complex systems and components within these systems (Figure 1.1). It permits the development of individual systems and components with clearly defined interfaces, data formats and communication protocols. This not only separates concerns and functionality for reusability, but also promotes pluggability and extensibility with uniform protocols and system/component life cycles. Instead of developing individual monolithic solutions for each science use case, the SoS architecture provides one solution that can be easily adapted to different use cases using different compositions of systems. It offers operational

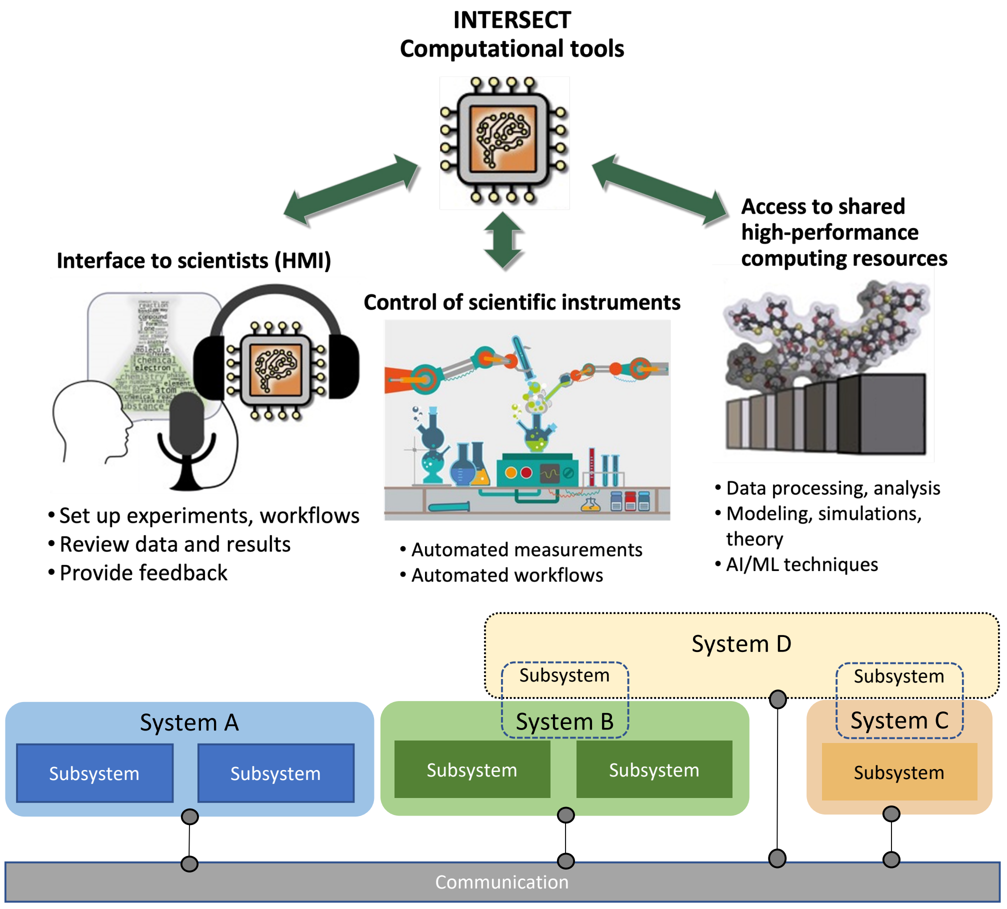

INTERSECT
Computational tools
Interface to scientists (HMI)
Control of scientific instruments
Access to shared high-performance computing resources
• Set up experiments, workflows
• Review data and results
• Provide feedback
• Automated measurements
• Automated workflows
• Data processing, analysis
• Modeling, simulations, theory
• AI/ML techniques
System D
Subsystem
Subsystem
System A
Subsystem
Subsystem
System B
Subsystem
Subsystem
System C
Subsystem
Communication

Figure 1.1. High-level illustration of INTERSECT tools with a System of System Architecture.

and managerial independence of systems and of components within systems, geographical distribution with a physically distributed and federated ecosystem, emergent behavior based on the interplay between systems and components, and evolutionary development through pluggability and extensibility.

1.2 PURPOSE OF THIS DOCUMENT

The SoS Architecture Specification document consists of various architectural views of the INTERSECT system that connects scientific instruments, robot-controlled laboratories, computational facilities, data centers, and scientific edge computing resources to enable autonomous experiments, "self-driving" laboratories, smart manufacturing, and AI-driven design, discovery, and evaluation.

This document incorporates the Insitute of Electrical and Electronics Engineers (IEEE) 42010 Standard entitled "Systems and software engineering – Architecture description." This architecture document uses the concepts of multiple, concurrent views to describe a complex system. The views are used to describe the system from the viewpoint of different stakeholders, such as end-users, developers, and project managers. There are many examples of view-based architectural descriptions such as "the 4+1 Architectural View Model" [3], the US Department of Defense Architecture Framework (DoDAF) [6], and the UK Ministry of Defence Architecture Framework [5]. We use a hybrid of the well-known 4+1 view model and the US DoDAF. The views used in this architecture specification are:

- *Logical view*: concerned with the functionality that the system provides to end users in the INTERSECT system.
- *Operational view*: concerned with the tasks and activities that support INTERSECT capabilities.
- *User view*: concerned with the human interaction needs within the INTERSECT system.
- *Data view*: concerned with the information needs in the INTERSECT system.
- *Physical view*: concerned with the engineering aspect of the underlying system components or resources.
- *Standards view*: concerned with standardization efforts across the INTERSECT system. The SoS Standards view also provides a block diagram to illustrate exactly where each standard impacts a given system.

The view model enables various stakeholders to establish the impact of the chosen architecture from their own perspective.

1.3 STAKEHOLDER REPRESENTATION

This section describes our main stakeholders and provides guidance on which sections of this document may be of direct interest to each stakeholders.

Stakeholders and Their Concerns

The architectural design of the INTERSECT SoS crosses many different groups, both within and outside of an organization. The following is a list of the stakeholder roles considered in the development of this architecture specification. For each role, the stakeholder concerns can be addressed by the views within this document. The following stakeholders shall be considered at a minimum:

- application software developers
- infrastructure software developers
- end users

- application and platform hardware engineers
- security engineers
- communications engineers
- system-of-system engineers
- chief engineer/chief scientist
- lead system integrator (LSI) program management
- system integration and test engineers
- external test agencies
- operational system managers
- maintainers
- representatives of standardization activities

Stakeholder Scenarios for Using this Document

For each role identified in Section 1.3, this section will contain a few short scenarios that explain how stakeholders in the role would use specific sections of the architecture document to help address their concerns.

Table 1.1. Stakeholder roles and the views within this document.

	View Chapters					
Role	**User**	**Data**	**Operational**	**Logical**	**Physical**	**Standards**
Application software developers	X	X		X		X
Infrastructure software developers		X	X		X	X
End users	X			X		
Application and platform hardware engineers						
Security Engineers		X	X		X	
Communications engineers		X				X
System-of-system engineers		X	X	X	X	X
Chief engineer/scientists	X	X		X		X
Lead System Integrator		X	X		X	
System Integration and test engineers	X	X	X	X		
External test agencies	X	X	X	X	X	
Operational system managers	X	X	X			

Maintainers	X		X			
Representatives of standardization activities		X	X			X

1.4 DOCUMENT SCOPE

In Scope

This Architecture Specification document covers the following aspects:

- The structure of components, their relations, and the principles and guidelines governing their design evolution over time [2].
- Technical framework for the systems comprising the SoS that designates how the systems will be used by the users in an operational setting, the internal/external relationships and dependencies between systems, components, and functions, and end-to-end functionality and data flow, including communications among the systems.

Out of Scope

This Architecture Specification document does not cover the following aspects:

- Cyber security policies.
- Data sharing and privacy policies.
- Specific technical software/hardware solutions.

1.5 DOCUMENT OVERVIEW

This Architecture Specification document is organized into the following sections:

Section 1 provides information about this document and its intended audience. It provides the purpose and document overview. Every reader who wishes to find information relevant to the architecture specification described in this document should begin by reading Section 1, which describes the purpose of this document, the stakeholders, an overview, and document control.

Section 2 provides the logical view and addresses the logical composition and interactions between the different systems in INTERSECT. This section includes system concepts, options, and capabilities.

Section 3 provides the operational view and describes the tasks, activities, procedures, information exchanges/flows from the perspective of the real-world operations stakeholders, like system administrators, maintenance, facility engineers and others.

Section 4 provides the user view and describes user roles and illustrates the different user interactions with the INTERSECT ecosystem. Wireframes are provided for communication purposes only.

Section 5 provides the data view and addresses the data needs, and describes the data framework that needs to exist to support the INTERSECT architecture. This section includes the overall data flow within the INTERSECT system, descriptions and definitions of data components, specification of the data messages for command and control plus data movement of bulk and streaming data, database schema for INTERSECT's operational data, and sequence diagrams to show data exchange.

Section 6 provides the physical view and describes the actual physical infrastructure at ORNL and its user facilities. Some parts of this section have been removed due to security concerns.

Section 7 provides the standards view and is currently being evolved. This view will identify specific standards that will augment the INTERSECT ecosystem.

An appendix contains wireframe diagrams, which illustrate user interactions with INTERSECT, and the XML schema for INTERSECT messages.

2 LOGICAL VIEW

2.1 INTRODUCTION

The Logical View will address the logical composition and interactions between the different systems in INTERSECT. In the Logical View, the operational and systems architecture models are linked together by describing how resources are structured and interact to realize the logical architecture specified in the Operational View (DoDAF references OV-2). The resource structure of a system is described in the logical view. The main purpose is to identify the primary sub-systems, performers (actors/agents) and activities (functions), and their interactions. This section of the architecture specification defines the Logical View, including an introduction to this view, System Concepts, System Options, and capabilities.

What does the Logical View contain?

- Definition of Architecture Concepts.
- Definition of System Options.
- System Resource Flow requirements capture.
- Capability integration planning.
- System integration management.
- Operational planning (capability and performer definition).

2.2 ARCHITECTURE OVERVIEW

INTERSECT is conceived as a system of systems architecture. In the logical view the focus is the interactions of the different systems that form an INTERSECT ecosystem. There are multiple ways to look at an ecosystem and depending on that viewpoint there are different systems or type of system. To address the different ways to view an ecosystem, we make a distinction between **Infrastructure Systems** and **Logical Systems**.

While **Infrastructure Systems** are tangible and supported by physical hardware, logical systems are conceptual only. **Logical Systems** do not directly map to a particular infrastructure system, but are the sum of the services provided by the **Infrastructure Systems**. They are separated by the functionality they provide. The logical view is introducing the different logical systems of the INTERSECT architecture, as well as the interactions between the logical systems.

Figure 2.1 shows the separation between logical and infrastructure systems. Depending on the view of the Systems, a logical system can have a number of infrastructure systems as it's subsystems, while a infrastructure system might have different logical systems as subsystem.

The INTERSECT architecture consists of the following logical systems:

- Infrastructure Management System
- Data Management System
- User Management System
- Campaign Management System
- Orchestration System
- Communication System

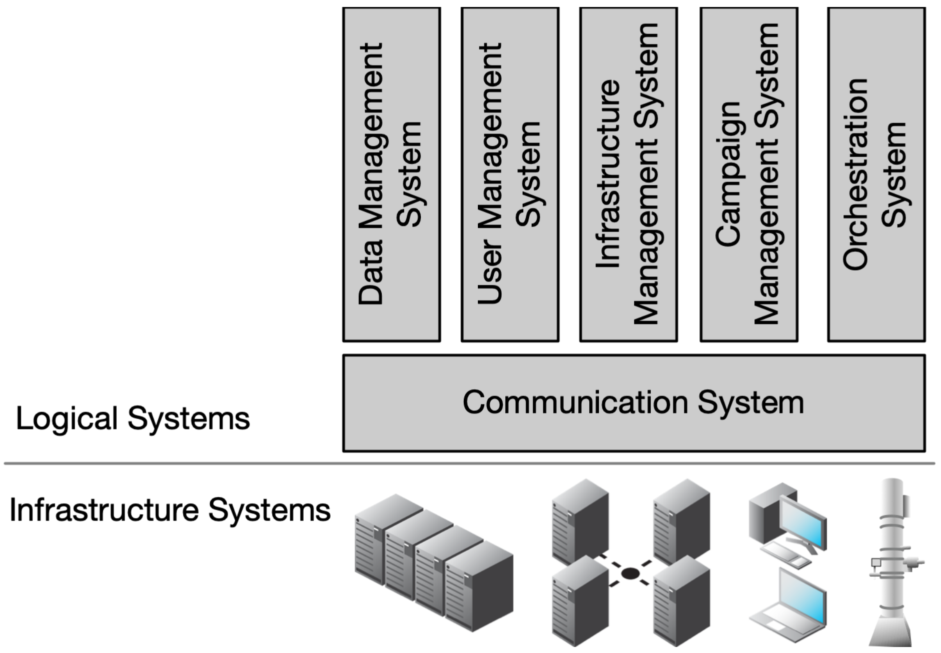


Figure 2.1. Differentiation of logical and infrastructure systems.

An INTERSECT ecosystem can be composed of many infrastructure systems, which provide services to other systems, the user, or both. An infrastructure system (i.e. an instrument control computer, or a virtual machine running on a physical computer) can contribute services to one or more of the logical systems.

Services provided by the different systems expose a set of capabilities. For example, the data management system must provide a capability to list available datasets. This capability can look different based on the infrastructure system providing the service. To take the instrument control computer as an example again, it must provide a capability to list all the datasets it is holding on locally accessible storage.

Figure 2.2 gives an overview of the services in the different logical systems.

The guiding principles for the INTERSECT architecture:

- System of Services
- Declarative composable Services
- Semantically composable Data

2.3 ARCHITECTURE CONCEPTS

INTERSECT provides an architectural framework to build ecosystems which make scientific workflows across different facilities more streamlined and transparent to the user. Additionally, INTERSECT makes it easier to cross scientific domains, bring experiment and simulation closer together, and provide an ecosystem where experimental data can be easily shared across different organizations.

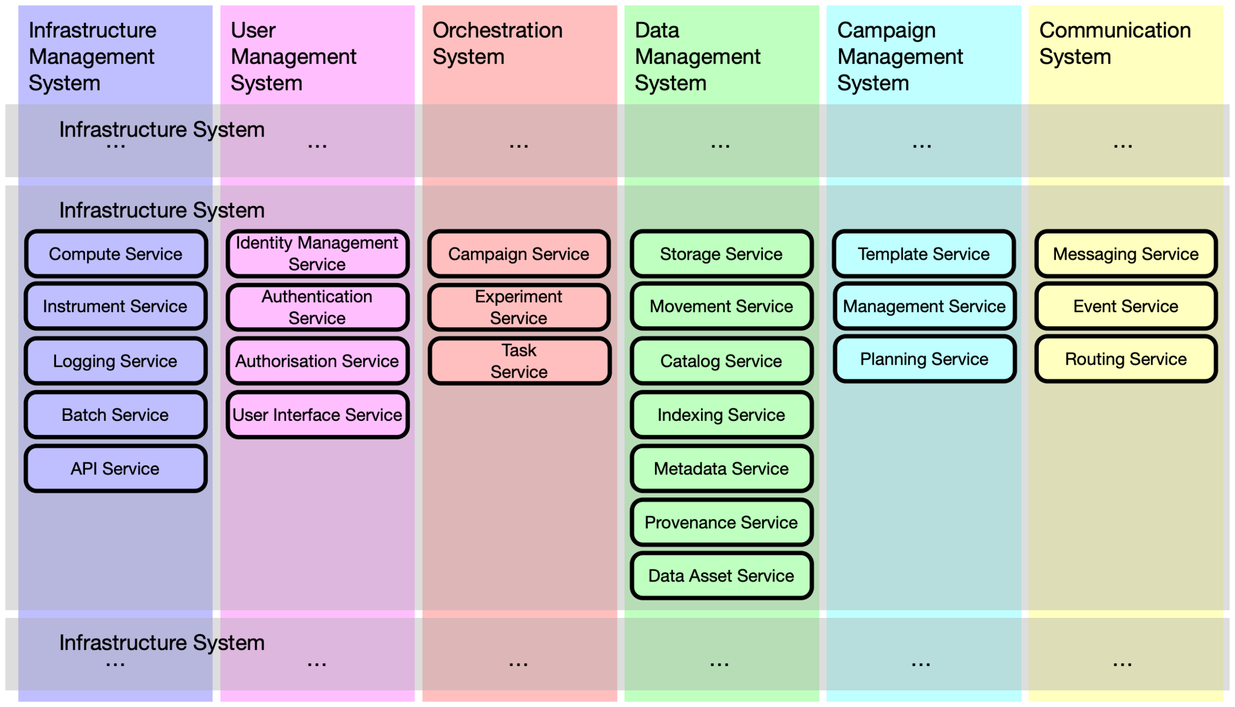


Figure 2.2. Logical Systems in the INTERSECT architecture

Experimental data [1] is one of the cornerstones of INTERSECT. This requires a Resource that can store high-level information of an experiment without the need to know the actual experimental data. This resource, we shall call it **Campaign**, must have a unique identifier. All additional data associated with a particular experiment must be linked with its campaign. Examples for additional data are: a particular workflow, any metadata for the experiment, provenance data, etc.

In many cases, access to **Instruments** is governed by program grants that provide a time allocation to a set of Instruments which is tied to a **Project** account. INTERSECT users can be members of a set of Projects, which in turn provides access to the Instruments in the projects scope. Instruments are utilized by **users** (Scientists, lab technicians, etc.) to run experiments and generate **Experimental Data**. Experimental Data can either be **stored** on a storage tier or **streamed** to another Instrument (Resource) for additional processing steps.

Instrument

A machine (or group of machines), with a specific capability. Typically a sample is observed by the instrument under controlled conditions. Some of these conditions are adjustable by parameters. The sample and the parameters are inputs for the instrument.

The output of the instrument is data of some kind in a specific format.

[1] Experimental data in this context means either inputs, outputs or both for an experiment, a simulation or analysis of raw experimental data.

Adapters

Synonyms: Connector, Controller, Agent

Adapters are providing interfaces between INTERSECT systems and external systems or services. An adapter provides a defined set of services and capabilities to interact with the external system. Additionally, it acts as a gateway to INTERSECT.

Instrument Adapter

Instrument adapters are specific to one instrument or a set of instruments. They represent the instrument digitally within INTERSECT. The instrument adapter must provide a description of the instrument to INTERSECT.

1) It must contain a list of the controllable parameters.
2) A description of the parameters, which must contain the input type of the parameter, the unit of the parameter and a valid range if applicable.
3) A list of the capabilities exposed to intersect.
4) A description of the capabilities.
5) A description of the output.

Instrument Controller

The instrument controller is an instantiation of the instrument adapter, a software agent with the ability to influence/observe/interact with the physical instrument it is attached to / it is controlling.

Service Adapter

Service adapters provide interfaces between INTERSECT and external services. Like instrument adapters, service adapters expose services specific to external services. Service adapters can be organized in service groups, which share similar capabilities and can be exposed by a more generalized interface. An example of this are authentication services, which share similar capabilities that INTERSECT is utilizing.

Authentication services can utilize different back-ends but expose the same interface to INTERSECT.

Intersect Systems and Services

INTERSECT defines a set of services. Generally, we differentiate between multiple types of services. Each service provides a set of capabilities to one of the logical Systems.

Naming and Addressing of systems and services

The Infrastructure management system relies on the premise that the infrastructure is going to be self describing. INTERSECT utilizes Uniform Resource Identifier (URI)s to name entities. For this document the scope for an intersect domain is *intersect.example.tld*. To address a system `<system>.intersect.example.tld` is used to address a system (either logical or infrastructure systems).

Example 2.1:

- `data.intersect.example.tld` for the data management system
- `orchestration.intersect.example.tld` for the orchestration system
- `infrastructure.intersect.example.tld` for the infrastructure system
- `turing.infrastructure.intersect.example.tld` for an infrastructure system named 'turing'

The URI `<system>.infrastructure.intersect.example.tld` can be used to access services or query an infrastructure system. A query to `infrastructure.intersect.example.tld` returns details about the infrastructure systems in this domain. To make application programming interface (API) calls to a particular system the fully qualified domain name (fqdn) is followed by the path for the API. For Example, to make an API call to a system named `turing` the following URI must be used:
`turing.infrastructure.intersect.example.tld/api/`

A service can have different return types based on the accept header of the HTTP request. For example, if a user opens an INTERSECT URI with a browser (i.e., the HTTP header "accept-header" is "html") a user interface shall be displayed or the request is redirected to the appropriate User Interface (UI). If the URI is not being opened with a browser, a machine readable object is returned to 'describe' the system.

We shall use HTTP(S)-based interfaces. The stack must implement all HTTP methods (`GET`, `PUT`, `POST`, `HEAD`, `PATCH`, `DELETE`, `OPTIONS`, `CONNECT`, `TRACE`) to interact with objects (or at a minimum the first 6 methods, `GET` to `DELETE`). Example 2.2 and 2.3 show how the initialization of the API shall look like. The `init` call does a `http get` to the URI and returns an object which describes the capabilities of that particular endpoint (using json, xml, etc.). Example 2.2 shows the initialization using a top level URI. Alternatively, as in Example 2.3, a client can be initialized to connect to a particular service or system.

Example 2.2:

```
client = Intersect.init('intersect.example.tld')
```

Example 2.3:

```
client = Intersect.init('compute.infrastructure.intersect.example.tld')
```

Using HTTP as the underlying mechanism, the INTERSECT ecosystem can utilize all the benefits of the web architecture like caching, authentication, etc. A URI can resolve to a static file describing the endpoint (i.e., the view of the compute infrastructure at the time of the request). In addition to URIs UUIDs shall be supported as selectors as well.

Example 2.4:

```
client.get_group([<res_1_uri>, ... , [<res_n_uri]).send(msg)
```

Example 2.5:

```
client.get_group([<res_1_uuid>, ... , [<res_n_uuid]).send(msg)
```

Example 2.4 and 2.5 show how to send a message to the group defined as parameters to the `get_group` function. While in Example 2.4 URIs are used to define the members of the group, Example 2.5 is using universally unique identifier (UUID)s. The selection of endpoints by capability must also be possible as shown in Example 2.6 and Example 2.7.

Example 2.6:

```
group = client.get_group('Intersect.Infrastructure.AllCompute')
```

Example 2.7:

```
group = client.get_group('compute.infrastructure.intersect.example.tld')
```

Another way to send messages to all ‘compute’ service endpoints, the infrastructure system can provide a broadcast capability under the `compute.infrastructure.intersect.example.tld/broadcast` endpoint.

Tasks, Commands and Actions

To abstract the activities within the INTERSECT ecosystem the terms `task`,`command` and `action` are used. Figure 2.3 shows the relationship between them. A `task` contains one or more commands. A `command` contains one or more actions. A `task` forms a graph of `commands` and `actions`, which are performed in series or parallel.

Figure 2.3 gives a more concrete example. The `task` ‘Analyze Data’ contains the `command` ‘Execute MPI Program’, which in turn contains the `action` ‘jsrun’.

Figure 2.4 also shows how the `action` for a `command` can be different between different infrastructure systems. In Figure 2.4 the `action` is ‘jsrun’ on System A and ‘mpirun’ on System B.

2.4 OBJECT MODEL

Entity

An `entity` or `object` is used to represent `resources` within INTERSECT. They are tuples consisting of a unique identifier (i.e., uri/iri, uuid, etc.) and a binary large object (blob), containing the (encoded) representation of the `entity`. Objects can link to other objects, creating some relationship between them.

Entity Types

All objects within INTERSECT have a type. Types have to be unique.

Some examples of object types are:

- `intersect.orchestrator.controller`
- `intersect.orchestrator.controller.campaign`
- `intersect.orchestrator.controller.experiment`
- `intersect.data.catalog`
- `intersect.data.set`

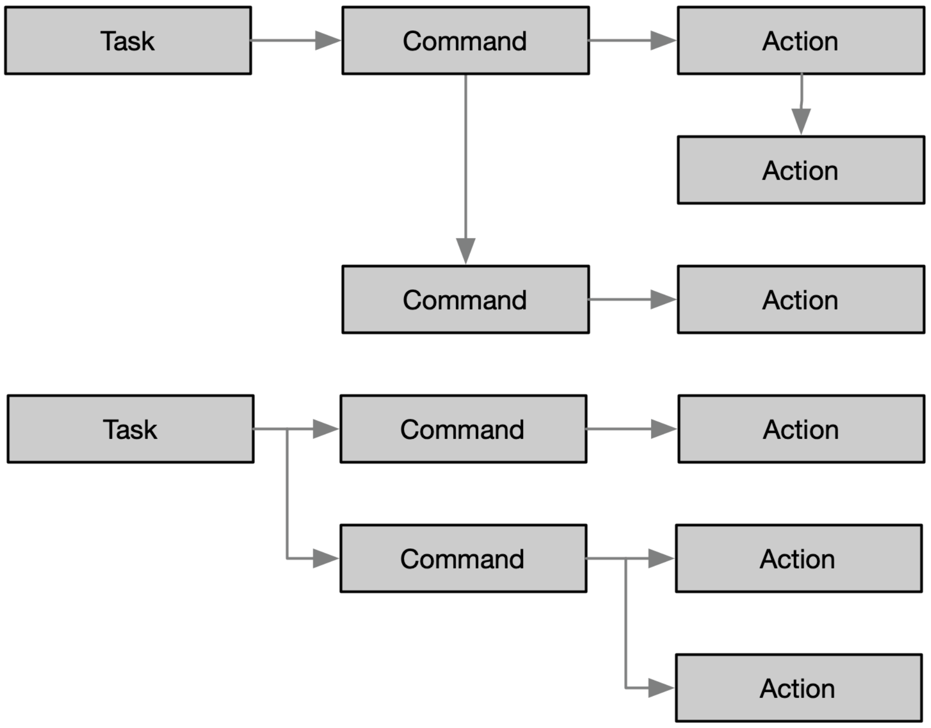


Figure 2.3. An abstract example of the relationship of task, command and action

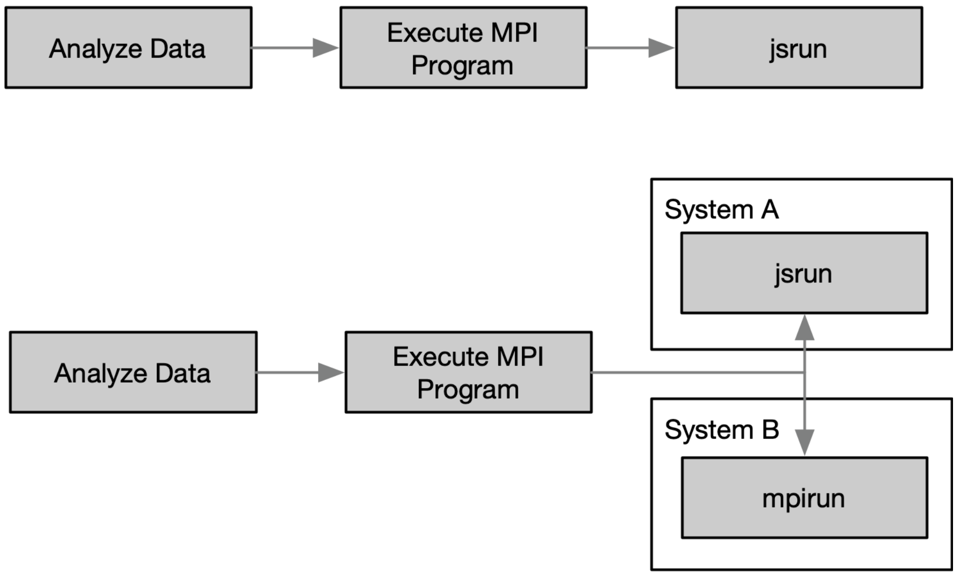


Figure 2.4. A concrete example of the relationship of task, command and action

- `intersect.data.asset`

Resource

Entities that are a distinct logical or physical component of a System. Resources most often represent components such as computers or scientific instruments that have particular sharing or partitioning constraints.

Capability

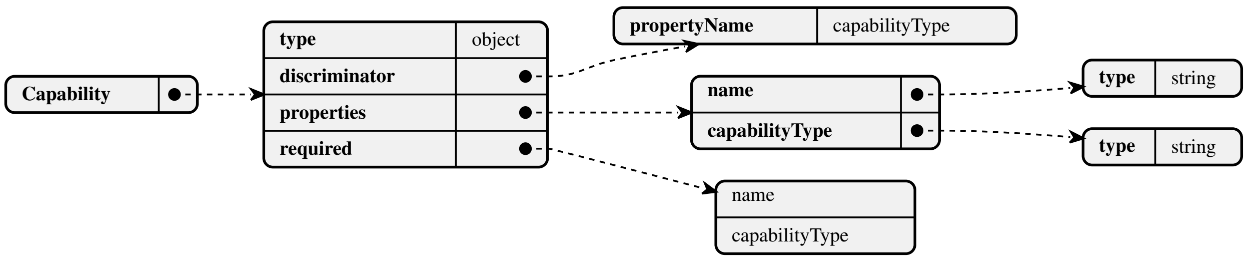


Figure 2.5. Representation of the capability object

A capability is the ability to perform an activity. It expresses the ability of an agent to perform an `activity` of a certain type. All capabilities must have a unique name. An agent can have multiple capabilities, and there multiple agents can provide the same capability.

Examples for INTERSECT capabilities are:

- (:intersect:compute:create)
- (:intersect:compute:submit)
- (:intersect:orchestrator:create)
- (:intersect:orchestrator:submit)
- (:intersect:orchestrator:start)
- (:intersect:data:create)
- (:intersect:data:get)
- (:intersect:data:list)
- (:intersect:infrastructure:create)
- (:intersect:infrastructure:register)
- (:intersect:infrastructure:find)

Example 2.8:

- Some agent has the capability to perform the activity described by an `entity` of a certain `type`.
- The agent *ORNL::Users::Some_User* wants to execute a bash script.
- The agent *Summit::Controller::Bash* has the capability to perform the activity *intersect::compute::execute::*

- The agent *Summit::Controller::Bash* performs activity *execute::script::bash* on behalf of agent *ORNL::Users::Doe::Some_User*

The bash script is an `entity` of type *script::bash*. To execute the bash script a controller with the capability *intersect::compute::execute::script::bash* is required. After a suitable agent is selected, it can perform the activity *intersect::compute::execute::script::bash*.

Agent

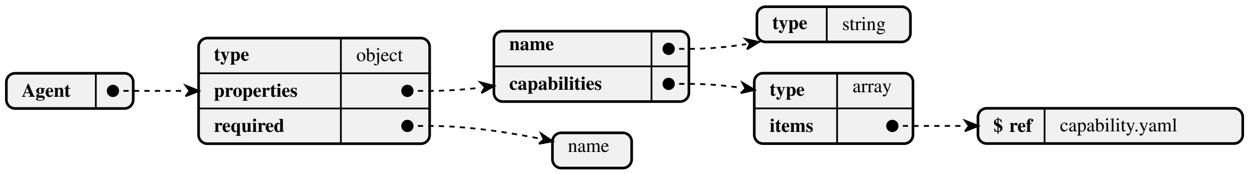


Figure 2.6. Representation of the Agent Object

An agent is a type of performer. An agent provides a capability. It can perform a specific type of **work**.

A **Controller** is a type of agent and provides capabilities to interact (control or query) with **Resources**.

Activity

An activity is a `unit` of `work`. It is an event, typically with a start and end, which is performed for a purpose. Activities can be repeated, and while the activity is the same, a separate event is generated each time a activity is performed.

Example 2.9:

The execution of a *program* is an activity, and every time the *program* is executed the activity is *performed* by an agent that has the capability *execute*

Properties

Properties are used to describe INTERSECT `objects` in more detail. For example, an infrastructure system providing a compute service must have a `property` for the processor architecture. Additionally, `properties` can describe other aspects of the infrastructure system, like memory, network, etc.

Constraint

Constraints are used to limit the resource selection to resources with certain properties.

Examples of constraints include:

- processor architecture
- memory
- accelerator

- number of CPUs / hardware threads
- number of nodes

Task

While an activity is a **unit** of **work**, a task represents the intend to perform an activity when certain conditions are met. The conditions can be dependencies and constraints.

A task has a set of `input` entities, blocks or consumes a `resource`, and produces a set of `output` entities. The `input` for a task can vary greatly and range from configuration parameters to physical samples. Resources that can be blocked are instruments that are time shared and in most cases consume some type allocations. Other types of consumables can be reactants or solvents. The ‘output’ of a task can vary greatly as well and range from reaction products, metadata about the reaction, to a publishable dataset.

A task has multiple states. It is either **ready** to be performed, **assigned** to a performer (as in **scheduled**), a performer is **active**ly performing the work, or the task has been performed either **successful** or **unsuccessful**. Additionally, it can also be **suspended**, **canceled** or **aborted**.

A task depends on the `capabilities` required by the associated `activity`. For example, assume the `activity` is to acquire the spectrum of a given sample, then the task depends on a spectrometer that provides this `capability`.

A task can have constraints. Constraints are used to express certain requirements of a given task. For example, a computational task can require a certain amount of memory, and constrains the task to systems with at least the required amount of memory. Other possible constrains are the central processing unit (CPU) architecture, a certain accelerator, number of nodes, etc.

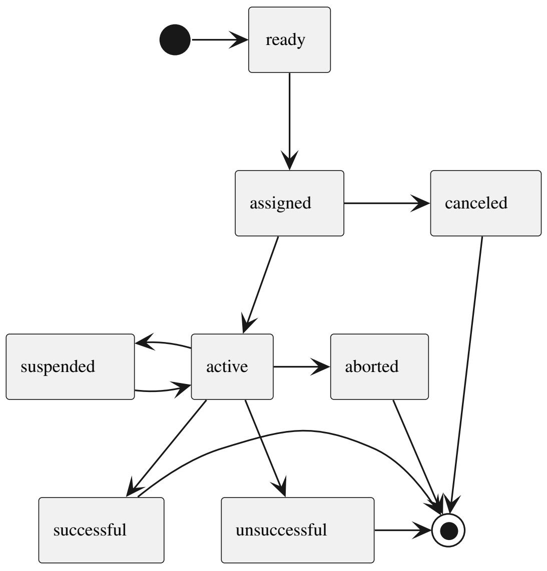


Figure 2.7. Task states

Example 2.10:

A user wants a bash script executed. To do so a task is created. The task will depend on:

- the bash script
- the capability to execute the script (i.e. an infrastructure system with a bash shell)
- any additional *constraints* or *dependencies*
  - *dependencies* like access to a particular resource (i.e. some input)
  - *constraints* like a minimum amount of memory or number of CPU cores

Once an agent that satisfies all *dependencies* and *constraints* (and policies?) is available, it will perform the activity of *executing* the bash script.

Example 2.11:

The **Microscope** has *capability* `measure` The **Instrument Controller** has capability `perform.measure.` The **Instrument Controller** has capability `configure.instrument`

A **Compute Controller** has capability `perform.analyze`

The **Experiment Controller** has the capability `perform.experiment`.

The **Experiment** requires the capability `perform.experiment`.

The first task in the experiment is `perform.measure`. This task is performed by the **Controller** with the capability `perform.measure`.

The second task is to analyze (`perform.analyze`) the output of the first task. This task is performed by some **Controller** with the capability `perform.analyze`.

Data Asset

Is a representation of a singular **Object / Entity**.

- as in files (*.jpg,*.svg, *.docx,*.sh)
- JSON Object

Data Set

Is a collection of **Data Assets** that belong together.

Metadata

Metadata objects are used to store additional information about objects. For example, a data asset can have metadata to provide more context.

Approval Point

A task which requires input of a specific type of agent. The execution of the task is suspended until the approval is either granted or denied.

Decision Point

A task whose outcome depends on the output of a previous one.

Template

A skeleton of an object, used as input to create other objects.

Event

The occurrence of an activity.

Experiment

An `experiment` is a collection of Tasks with the goal to achieve a specific outcome. It contains a description of the data flow between the different Tasks and the dependencies between them.

An experiment has at least one `controller`, which coordinates the execution of the `experiment`.

An experiment has one template

An experiment can have a dependency on:

- Task / Activity
- Performer / Agent / Controller
  - Types of Performers
- Event
- Capability
- Campaign
- Experiment
- Data Set
- Data Asset
- Approval Point
- Decision Point
- Template

Example 2.12:

An experiment can depend on instruments or resources. The example experiment depends on a microscope, and can utilize its capabilities. Additionally, it contain a task that takes the output of the microscope to do some post-processing. The task itself has a dependency on a compute service. To run the experiment, the microscope needs to be configured. The microscope creates data for the output.

Campaign

A campaign manages the state of, the data flow between and the dependencies of a non-empty set of experiments.

A campaign has a controller

A campaign has a template

2.5 SYSTEM DESCRIPTIONS

Infrastructure Management System

[!uri] **infrastructure.intersect.example.tld**

The Infrastructure Management System (IMS) is responsible to manage infrastructure systems (systems are functional elements, i.e., a HPC system, a Microscope, etc.) that are part of INTERSECT.

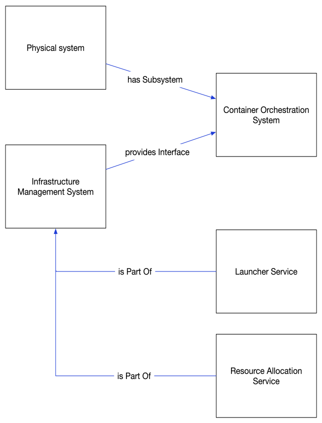

Physical system
has Subsystem
Container Orchestration System
provides Interface
Infrastructure Management System
is Part Of
Launcher Service
is Part Of
Resource Allocation Service

Figure 2.8. Relationship of different systems to provide a container as a compute capability

Example 2.13:

Manage Compute resources on a container platform

The IMS on a container orchestration system consists of a resource allocation service and a Launcher service.

In this scenario the Resource Allocation Service can deploy containers. The Launcher service can start services in the container platform. Containers deployed might hook into the launcher service, to run *task* in a container.

Example 2.14:

Manage Compute resources on a batch job platform

The IMS on a container orchestration system consists of a *resource allocation service* and a *launcher service*.

In this scenario the Resource Allocation Service allocates compute resources using the batch system (i.e. submitting a job). It `may` provide functionality to create reservations. The Launcher service *might* become available once the requested resources are available.

Node Management Service

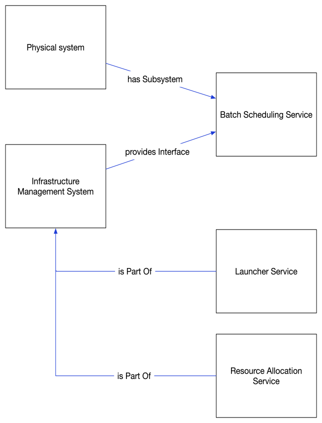


Figure 2.9. Relationship of different systems to provide a batch job as a compute capability

The node management service is responsible to announce the nodes capabilities. Additionally, it is responsible to ensure that the node state satisfies the nodes configuration. Furthermore, it announces the nodes status to other interested agents.

Launcher Service

[!uri] **launcher.infrastructure.intersect.example.tld**

The Launcher Service is responsible of starting up instances of Tasks, Services and Systems. The service listens for 'Launch' events and start an instance of the required entity.

Types of launchers:

Launch Task Launch a 'Task'; Tasks are executables that run on a compute capability

Launch Service Launch a 'Service'; Services are provided by INTERSECT (i.e., Orchestrator, etc.)

Launch Infrastructure System Launch a 'System'; Systems are entities that provide Services to the infrastructure (shared) or a Project or campaign (dedicated)

Resource Allocation Service

[!uri] **allocate.infrastructure.intersect.example.tld**

Request and allocate resources. Resources can come from batch scheduled (HPC) Systems, on- or off-premises cloud resources, or any other mechanism that can dynamically allocate resources (i.e. function as a service).

There are multiple ways to allocate these resources: Submit a **job** and

- start an INTERSECT agent/service
- start a user defined task
- start a predefined **workstep**

In the first case, INTERSECT is able to add the nodes as resources into the intersect environment and control the **compute resources** within the job allocation.

Instrument Controller Service

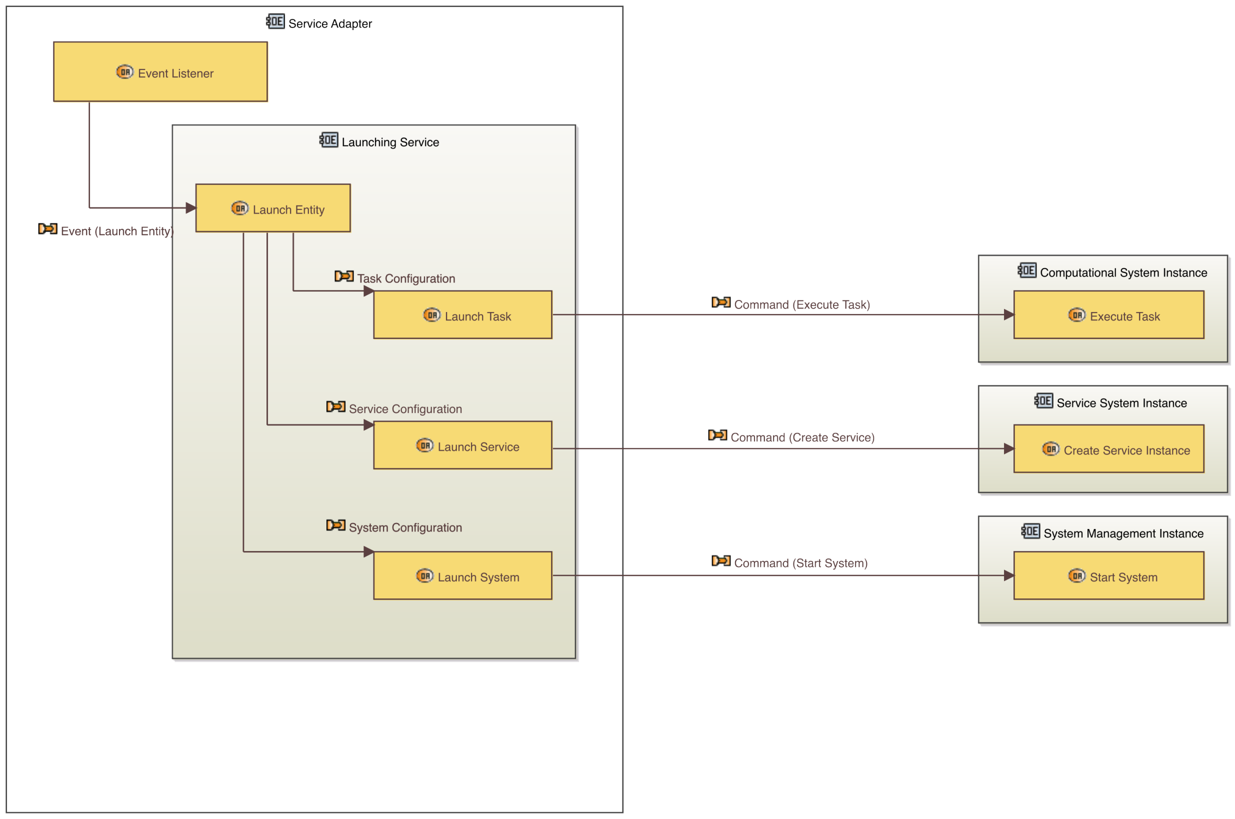


Figure 2.10. General capabilities of the Launching Service.

[!URI] **instrument_name.infrastructure.intersect.example.tld** or **infrastructure.intersect.example.tld/**

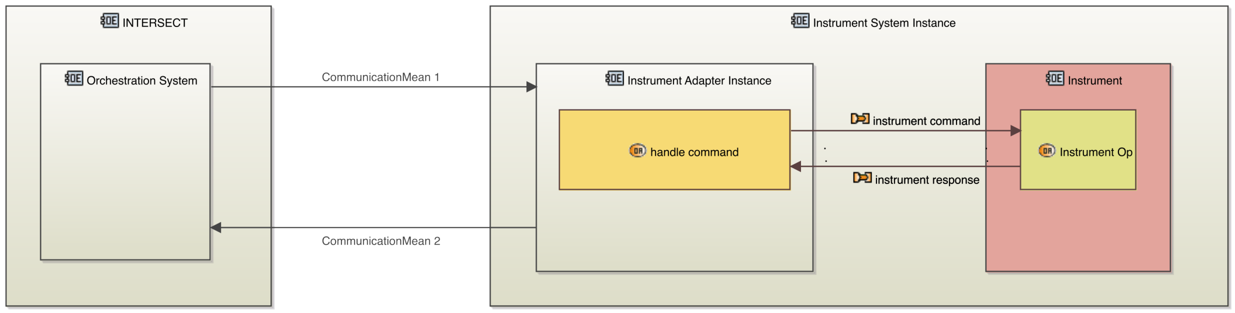


Figure 2.11. Instrument adapter capabilities.

Figure 2.11 depicts common capabilities for the instrument adapter, which includes:

- Provide Instrument Description
  - Instrument capabilities
  - configuration options
  - metadata
  - data formats
- Provide Status Information
- Respond to Requests from INTERSECT (orchestrator)

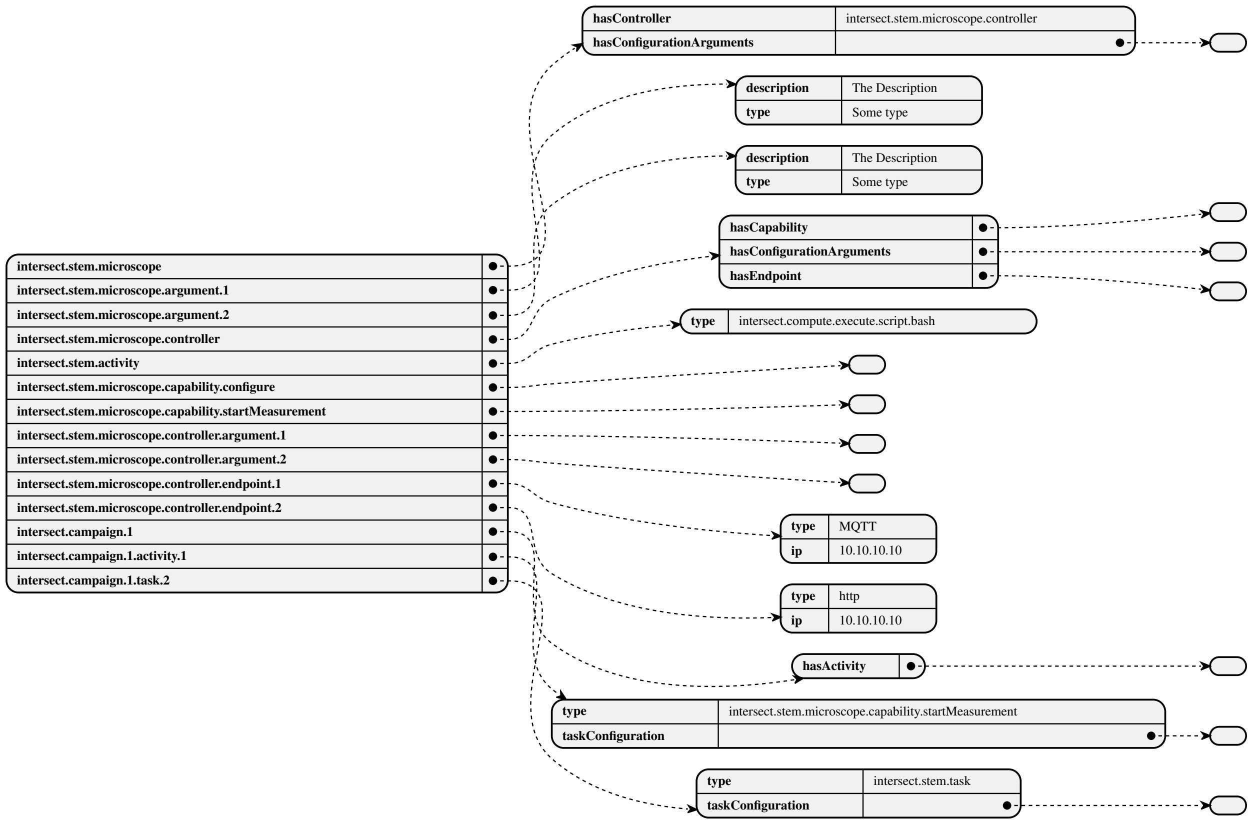
Provide Instrument Description

Capabilities for instruments are specialized and highly dependent on the type of instrument. To support a rich set of instruments, the set of capabilities for each instrument needs to be advertised to INTERSECT. To facilitate this, an instrument has to be described. The instrument description has to contain the description of the parameters in some form (I.e., parameter name, value range, etc.).

Capabilities must either be well defined, follow an ontology, or a schema to describe the capabilities and the corresponding commands to execute them.

- Command arguments must be described.
  - Name and description of the command
  - mandatory and optional parameters
  - type of the parameter (i.e. bool, int, string, etc.)
- Configuration parameters must be described.
  - Name and description of the configuration parameters
  - mandatory and optional parameters
  - default values
  - type of the parameter (i.e. bool, int, string, etc.)
- Metadata of the instrument must be described.
  - Name and description of the metadata fields
  - mandatory and optional metadata
  - default values
  - type of the metadata field (i.e. bool, int, string, etc.)

- Data Asset format must (shall?) be described



hasController
intersect.stem.microscope.controller
hasConfigurationArguments
description
The Description
type
Some type
description
The Description
type
Some type
hasCapability
hasConfigurationArguments
hasEndpoint
type
intersect.compute.execute.script.bash
intersect.stem.microscope
intersect.stem.microscope.argument.1
intersect.stem.microscope.argument.2
intersect.stem.microscope.controller
intersect.stem.activity
intersect.stem.microscope.capability.configure
intersect.stem.microscope.capability.startMeasurement
intersect.stem.microscope.controller.argument.1
intersect.stem.microscope.controller.argument.2
intersect.stem.microscope.controller.endpoint.1
intersect.stem.microscope.controller.endpoint.2
intersect.campaign.1
intersect.campaign.1.activity.1
intersect.campaign.1.task.2
type
MQTT
ip
10.10.10.10
type
http
ip
10.10.10.10
hasActivity
type
intersect.stem.microscope.capability.startMeasurement
taskConfiguration
type
intersect.stem.task
taskConfiguration

Figure 2.12. Example json document describing an instrument

In the case of LabOS[2], the experiment workflow could be one parameter. Additionally, we need a way to communicate the current state of the experiment. This might happen through messages (if LabOS provides these capabilities) or polling.

Key and token management

Provides capabilities to generate authentication and authorization tokens.

Data Management System

The Data Management System (DMS) has already been mentioned. It is responsible of tracking the input and output data of a campaign. The DMS is distributed by nature and all entities providing storage capabilities to INTERSECT become part of it.

Figure 2.13 gives a broad overview about Data Management as a discipline. INTERSECT is focused on **Data Integration** and **Data Service**. **Data Integration** is about combining data from different sources and

[2]We currently don't know how the communication with LabOS will look like, but the assumption is that we will send a workflow description for the experiment to LabOS.

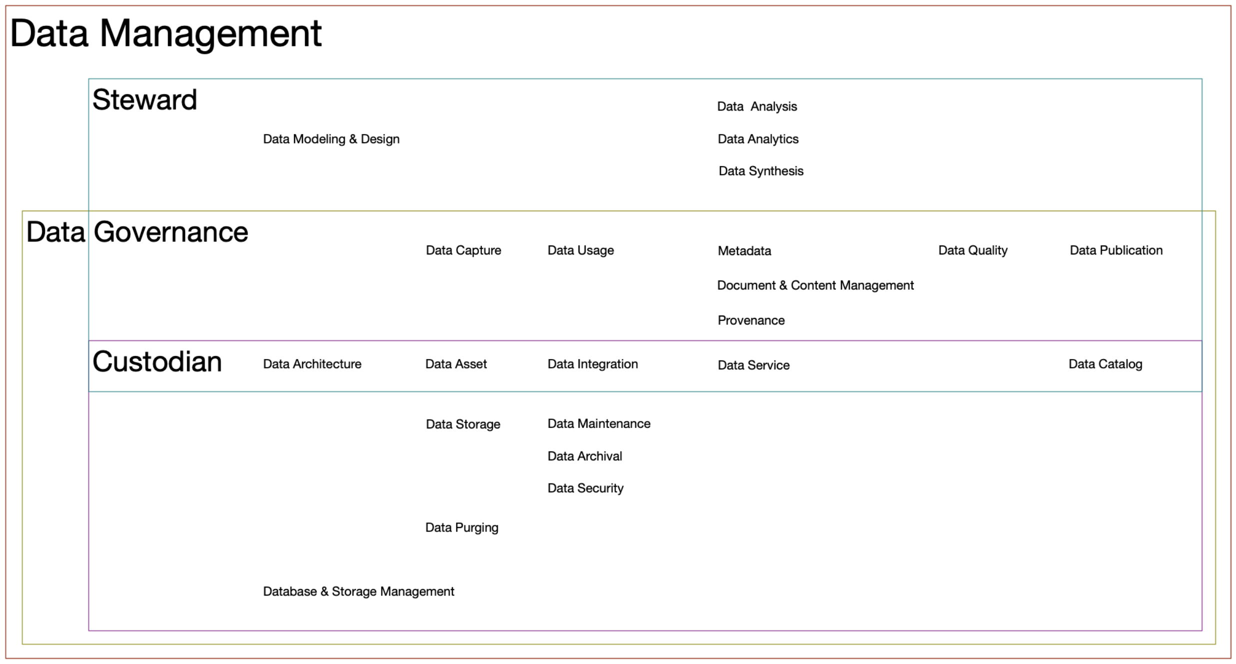


Figure 2.13. Overview of Data Management

to provide a unified view of the data. A **Data Service** provides one or more capabilities to interact with (certain types of) data. INTERSECT provides the ideal platform to implement a variety of **Data Services** on top of the unified view the **Data Integration** provides.

A **Data Service** provides capabilities that operate on one or more **Data Assets**. Some basic capabilities INTERSECT must provide are:

- finding the physical location of a particular **Data Asset**
- move a **Data Asset** to a different physical location
- provide a registry / catalog of **Data Assets**

A **Data Asset** is an abstraction of domain specific data. It has a unique identifier. The DMS provides services to move data.

Task: Move the **Data Asset** `sns.ornl.gov/data/aae1af09-ece6-4f6d-9bf2-7ce97997c3f0` to `data.olcf.ornl.gov`

The idea is that an infrastructure system provides a filesystem that is going to be exposed to intersect.

- The **DMS** *defines* the **Interfaces**.
- **(Micro) Services** *implement* the **Interface(s)**
- The **(Micro) Services** may *run* / may be *executed* on the same, or a different Physical System

Indexing Service

Indexing services are used to index data within a given scope. The indices facilitate finding data of interest. A scope is typically represented by a data catalog.

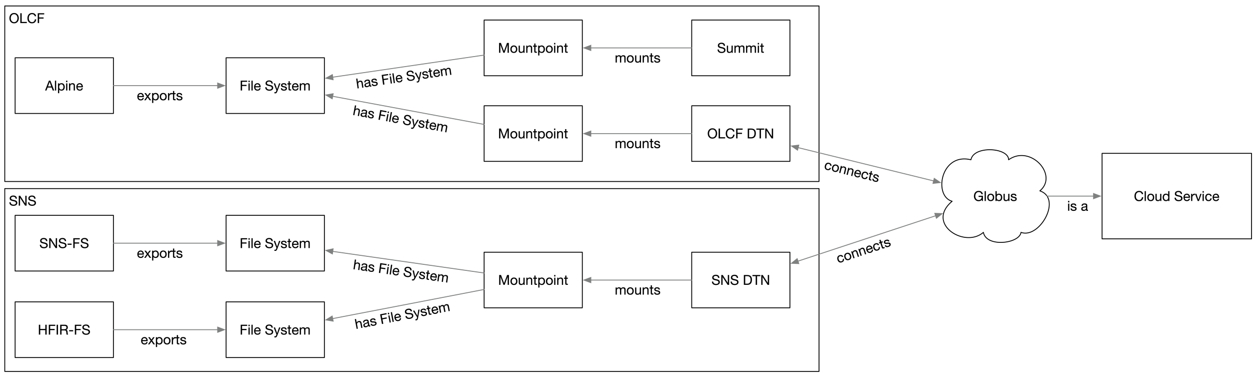


Figure 2.14. File System

Data Catalog Service

A data catalog service provides a catalog of data sets available within a given storage domain. Catalog services provides abstract representations of the data and its metadata.

Data Transfer Service

Data transfer services facilitate movement of data between storage domains.

Metadata Service

Storage, indexing, and capture of metadata.

User Management System

The User Management System (UMS) is responsible for managing INTERSECT users and their corresponding profiles and settings.

Additionally, it is responsible for authentication and authorization.

User Agents

User Agents are entities that represent a user for automation purposes.

Campaign Management System

The Campaign Management System (CMS) manages campaign templates and recipes.

Orchestration System

The orchestration system facilitates the interaction between the different actors (Instrument Adapters, Compute services, etc.) that are part of a campaign.

A service instance is started by an infrastructure service (i.e., launch service).

Alternatively, a user may start an orchestrator instance manually.

If the orchestration system manages the creation of the orchestrator instance, the startup is triggered by an event (command).

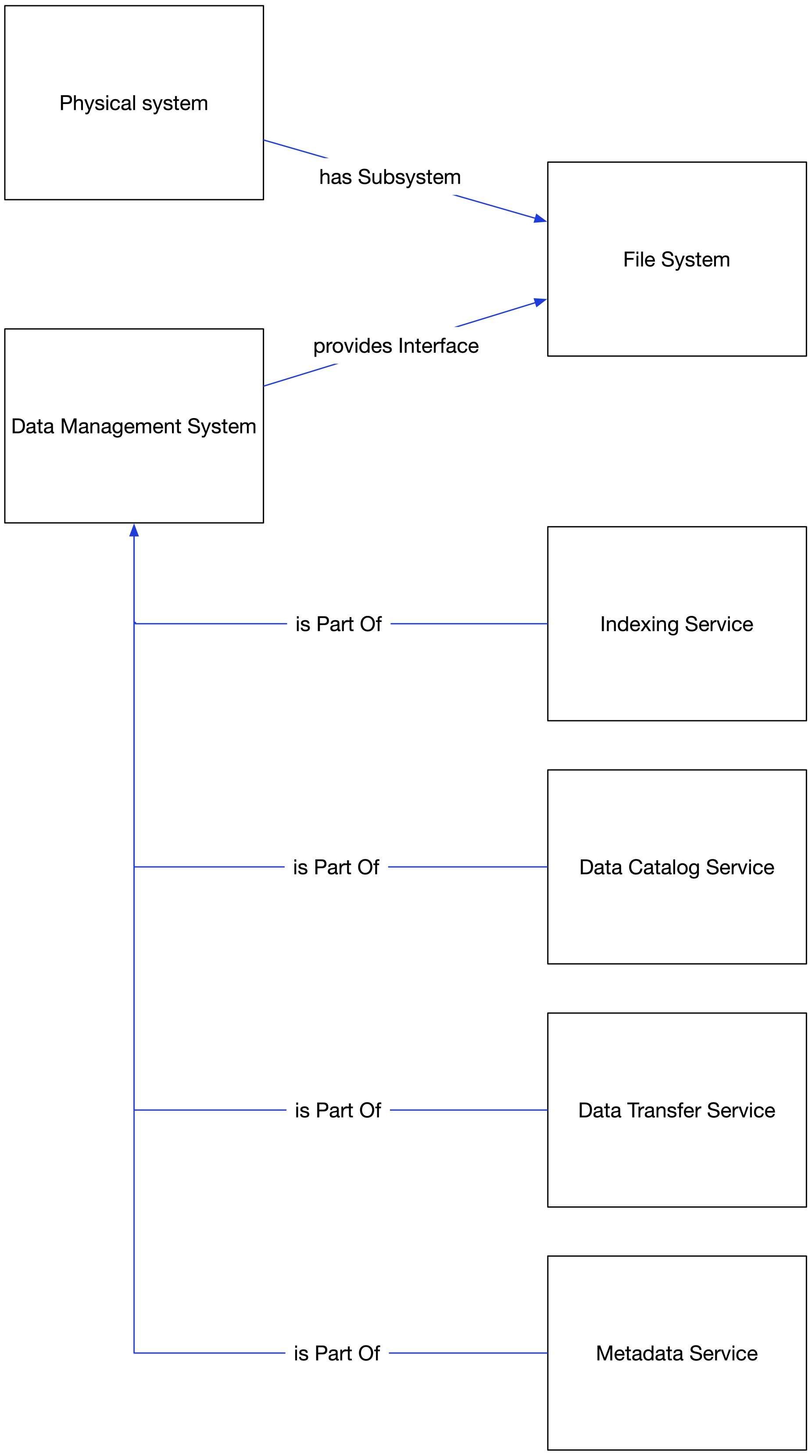


Figure 2.15. Data Managment

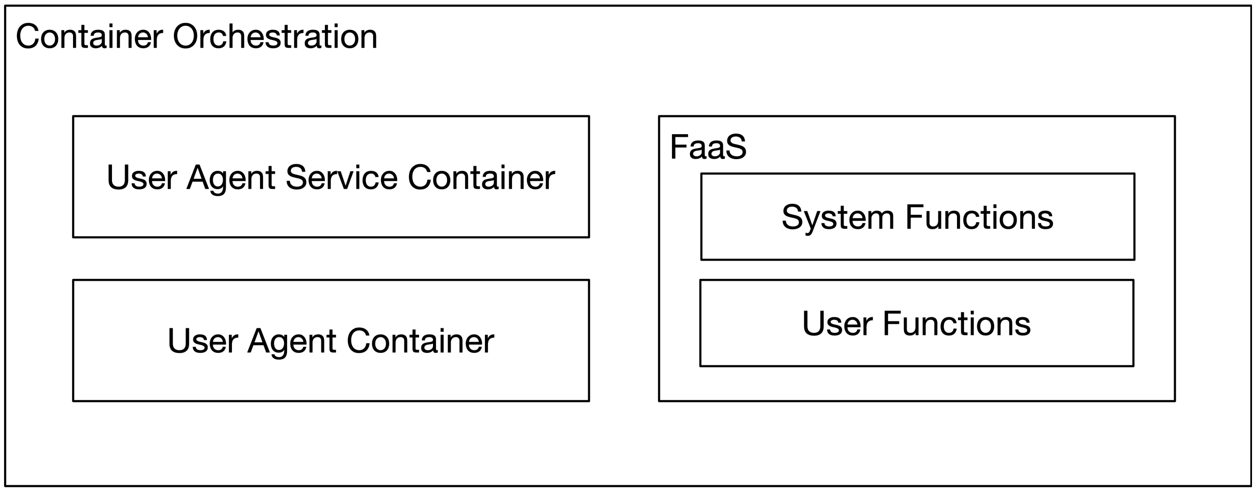


Figure 2.16. Container Orchestration

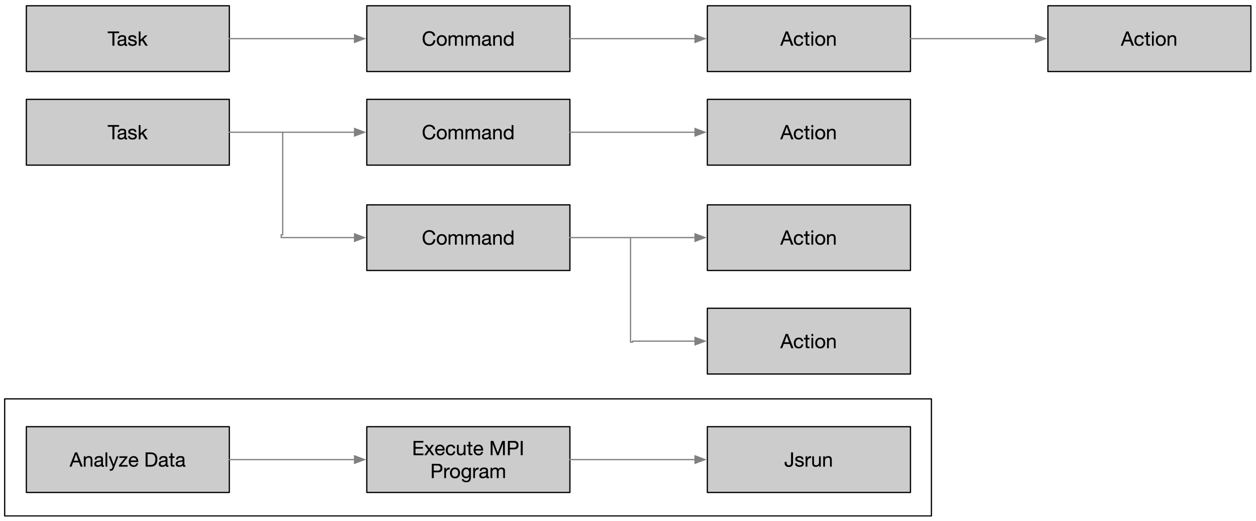


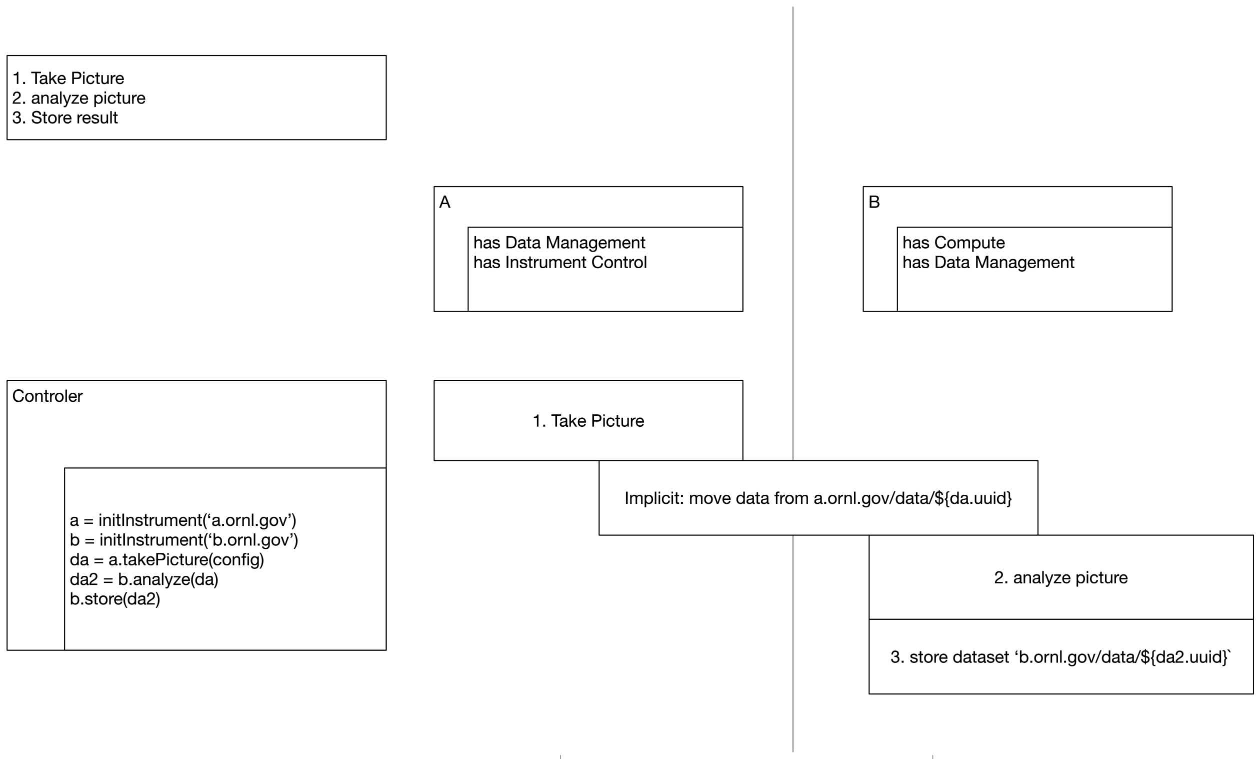
Figure 2.17. Tasks

The event can come from a scheduling service or through the HMI.

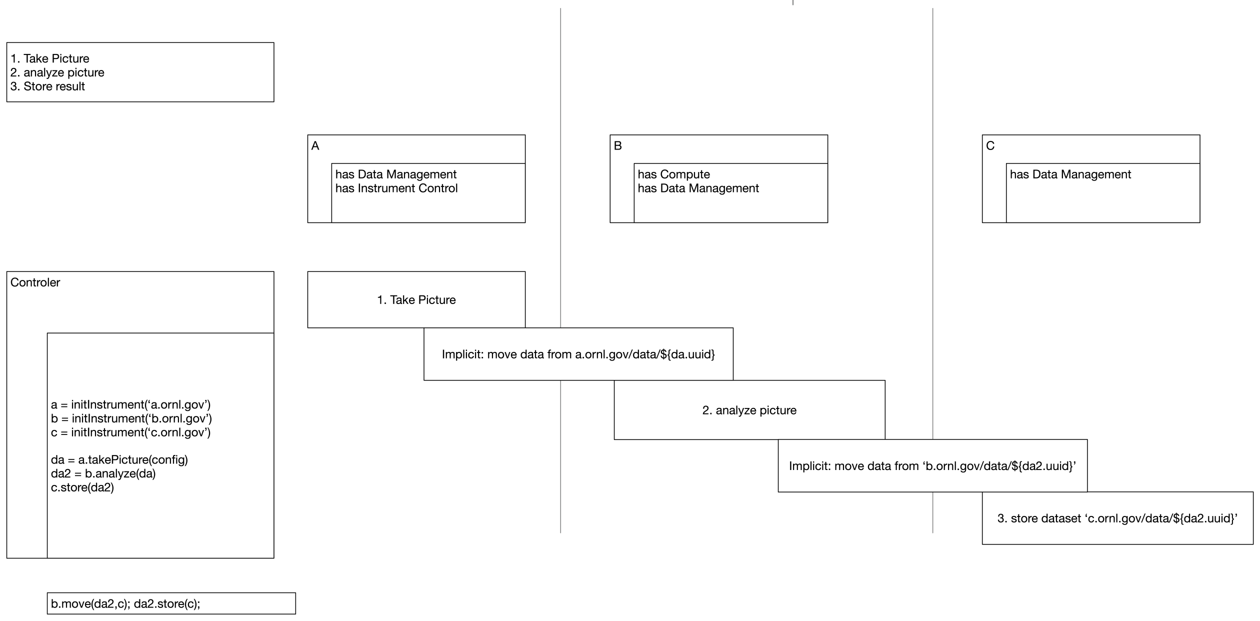
During the creation of the campaign environment, the orchestrator will connect to the individual instruments and services and configure them for the experiments, or start services required by the campaign if they do not already exist.

```
a = initInstrument('a.infrastructure.intersect.example.tld')
b = initInstrument('b.infrastructure.intersect.example.tld')
c = initInstrument('c.infrastructure.intersect.example.tld')

da = a.takePicture(config)
% intersect.move(from: da.filepath, to: b)
da2 = b.analyze(da)
% intersect.move(from: da2.filepath, to: c)
c.store(da2)
```



```
a = initInstrument(‘a.ornl.gov’)
b = initInstrument(‘b.ornl.gov’)
da = a.takePicture(config)
da2 = b.analyze(da)
b.store(da2)
```



```
a = initInstrument(‘a.ornl.gov’)
b = initInstrument(‘b.ornl.gov’)
c = initInstrument(‘c.ornl.gov’)

da = a.takePicture(config)
da2 = b.analyze(da)
c.store(da2)
```

```
b.move(da2,c); da2.store(c);
```

Orchestrator

An Orchestrator is a dynamic service with (at least) one service instance for each campaign.

The orchestrator is controlling the execution of the campaign, or a subset of the campaign. It can be compared to the controller in the design pattern document.

Once the service is started up, it will initialize the campaign. Initialization consists of fetching the campaign configuration () and creation of the campaign environment [3].

Orchestrator Instance States:

Initializing The Orchestrator is initializing the Campaign Environment
Running(?) The Campaign is running, and the Orchestrator is busy
Finalizing The Orchestrator is cleaning up the Campaign Environment

Communication System

The Communication System provides the means for the different nodes, systems and services to communicate.

Event Broker Service

The event broker enables INTERSECT (sub)systems to act on events in the system. Adapters can subscribe to events and publish events themselves. This might require some restrictions as to what events adapters can subscribe to, what type of events it can publish/or how it can publish events, and limits on.

Events *must* to be defined:

- INTERSECT will have pre-defined events.
- INTERSECT compliant systems *shall* be able to define events
- INTERSECT compliant services *shall* be able to *listen* to a non-empty set of events.
- INTERSECT compliant services *shall* be able to *emit* a non-empty set of events.

Events *(must/shall)?* have a scope:

- scoping can happen by
  - System
  - Jurisdiction
  - Facility
  - ?

Message Broker Service

The message broker is responsible to route messages to the correct recipient.

A message can have multiple recipients.

Messages *must* to be defined:

- INTERSECT will have pre-defined messages.
- INTERSECT compliant systems *shall* be able to define messages
- INTERSECT compliant services *shall* be able to *receive* to a non-empty set of messages.
- INTERSECT compliant services *shall* be able to *send* a non-empty set of messages.

Broadcast Messages *(must/shall)?* have a scope:

- scoping can happen by

[3]The campaign environment contains all the services utilized during a campaign. Creating the environment will either start the required services or connect to the service if it already exists.

  - System
  - Jurisdiction
  - Facility
  - ?

2.6 SYSTEM OPTIONS

Orchestration

- Local orchestration
  - Orchestrator is running on a local resource (i.e., the instrument control computer)
- Remote orchestration
  - The orchestrator is running remote (i.e., in a hosted [4] container)
- Distributed?
  - Multiple orchestrators coordinate a campaign
- Automated
  - A campaign is scheduled for execution and the experiment plan is executed automatically
- Interactive
  - The campaign is started, and possibly controlled, by a scientist or operator

Execution modes

- Container
  - The System/Service/Capability is executed in a containerized environment
- Native
  - The System/Service/Capability is executed natively
  - Native executable in a system process, as a system service
  - Native executable as a script (bash, python, etc.)

Data acquisition modes

- Streaming data
  - Multicast?
- Buffered data
  - Memory
  - File
    - Local
    - Remote

[4] Hosted in a cloud resource, vs. a container on a local resource (The orchestrator could run in a container environment everywhere).

3 OPERATIONAL VIEW

This section of the architecture specification defines the Operational View, including an introduction to this view, a high-level view of systems and sub-systems, and different activities that are relevant from an operational perspective.

3.1 INTRODUCTION

This section introduces the operational view, what is included and not included in this view, and stakeholders and concerns.

Operational View Definition

The Operational Viewpoint describes the tasks, activities, procedures, information exchanges/flows from the perspective of the real-world operations stakeholders (i.e., systems administrators, maintenance, facility engineers, system managers, instrument scientists).

Ideally the Operational View will not be specific to a particular resource (e.g., HFIR CG-1D instrument, OLCF Summit), but in practice will reflect facility constraints and procedures. These organizational restrictions influence the current operational procedures (operations). These restrictions may have areas that could be improved/streamlined if other aspects of the system architecture change (i.e., changes to Physical Model could impact Operational View).

Definition: "The Operational Viewpoint describes operational scenarios, activities, and requirements that support capabilities." [6, pg.68]

What is in the operational view?

The Operational View captures the perspective of people, machines, tasks, policies for the system (i.e., primarily aspects associated with Operators & Administrators), to include:

- Creating the connections (networking) for producer/consumer flows.
- Monitoring metrics for information exchange, which will differ depending on the type of information and entities, i.e., human/human exchanges (email), device/device (packets). This also include monitoring of system resources, e.g., resource utilization, load imbalances, etc. (cf: Data View for metrics data)
- Hierarchy of responsibilities (system/network/instrument administrators)
- Policies and procedures (e.g., steps for adding new hardware to a lab space, steps for adding (and restricting) users to (from) resources)

What is NOT in the operational view?

The Operational View is not concerned with things like the following:

- Not concerned with details of messaging formats for data exchange (i.e., message contents, with possible exception for security monitoring)
- Not concerned with details of an individual user application (i.e., interested in supporting/enabling all users of the resource)

Stakeholder and concerns

The primary stakeholders for the Operational View are the Operators and Administrators (see Table 4.1), which enable Users to take advantage of the Resources made available by their Owners. The Owners are concerned with having the resources used effectively by the Users. It is the Operators and Administrators that facilitate this capability. The Operational View reflects these aspects.

Advice to Users: A single person may fulfill one or many of roles. For example, a system software developer may also perform administration for the resources being managed via the software. In contrast, these roles may be split among different individuals. A key distinction is the level of details and operational responsibilities. The documentation and interfaces are likely different to accommodate the interests of these different roles.

3.2 HIGH-LEVEL OPERATIONAL DIAGRAM

For understanding, Figure 3.1 depicts an example of an operational view of a system. In this figure, two different facilities are shown – Center for Nanophase Materials Sciences (CNMS) and Oak Ridge Leadership Computing Facility (OLCF) – both located at ORNL and participants of INTERSECT. Even though they are located at ORNL, they are administered and operated by different individuals. These different Operators and Administrators can only see a section of INTERSECT (aka, their own facility and/or resource).

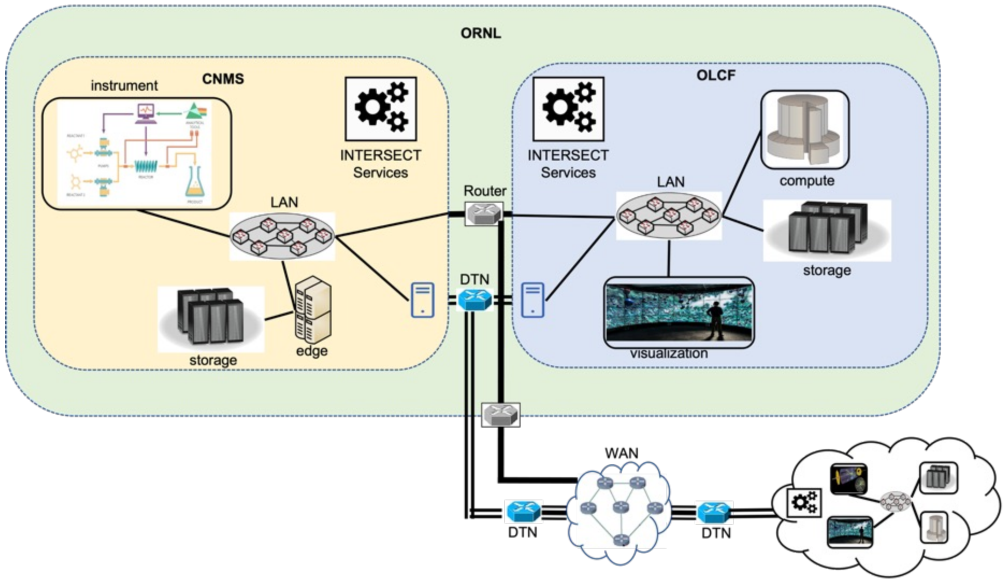


Figure 3.1. High-level diagram to assist operational view of system; showing different facility domains that will be managed by different Operators and Administrators depending on the resource and Facility.

High-level view of Systems & Sub-systems

The high-level view show in Figure 3.2 is intended to give a succinct depiction of the overall System-of-Systems Architecture. The subsequent views define details for this high-level view from the perspective of various stakeholders. The high-level view provides a common reference point for key elements that are relevant for all views.

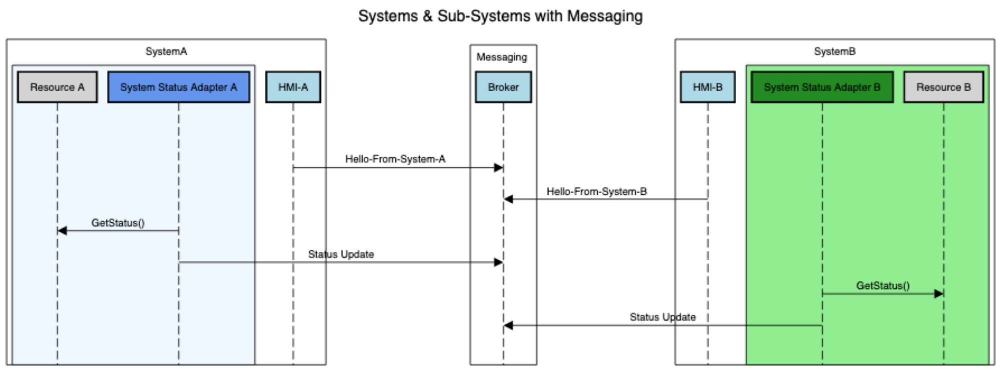


Figure 3.2. High-Level View of Systems & Sub-systems High-Level View of Systems & Sub-systems with a brokered messaging paradigm for communication.

A System is a group of interacting, or interdependent elements, that provides a defined functionality. These systems are comprised of Sub-system that provide useful capabilities to one or more systems, and ultimately provide the necessary functionality for the overall system.

A Service is the mechanism that provides access to one or more of the capabilities of a given system. A service has a software interface that follows well defined properties, i.e., software contracts.

An INTERSECT enabled environment is one that honors the defined Service contracts to expose the Systems. In the reference implementation, this SoS architecture is based on micro-services. Ideally, these micro-services should limit coupling where possible and hide underlying heterogeneity of resources and implementation details from the user.

There is a shared messaging layer among the services. This messaging layer should provide a basis to bootstrap services, inter-service communication, limit scope/visibility on messages, honor message durability, ordering and resilience requirements. The Messaging layer should also support both subscription (“push”) and direct (“pull”) based messaging patterns.

Advice to Implementors: As with other parts of the overall design, the full details on Messaging requirements are still in-progress; however, it does appear that publish/subscribe and point-to-point messaging will be beneficial.

Orchestrators and Commands

An Orchestrator is used to manage logical steps (operations) for a particular series of actions. Since the series of actions may be distributed over different resources and services with different levels of complexity, the Orchestrator is the intentiy that can be used to coordinate the actions. An INTERSECT *Command* translates a generic operation into a target specific *Action*, i.e., a generic Command is mapped to backend specific Actions. A *Command-Set* is a group of *Commands*[5]. Some services, like an Orchestrator, may choose to translate Command(s) into new Command(s), e.g., transform to improve performance and/or reliability. Additionally, an Orchestrator may choose to pass Command/Command-Sets to other Orchestrators in the ecosystem for assisting in the execution of operations.

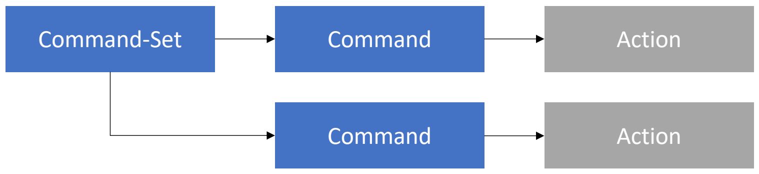


Figure 3.3. Diagram showing hierarchy of Actions, Commands and Command-Sets.

For example, consider a generic "analyze data" procedure that includes the execution of a MPI program that subsequently uses a launcher (e.g., `jsrun`) to perform the compute actions. This sequence of $AnalyzeData \rightarrow RunMPIprogram \rightarrow jsrun$ would map to $CommandSet \rightarrow Command \rightarrow Action$, where the final action is specific to the underlying compute system that uses a jsrun-based launcher.

Example: Cross Facility Interaction for CADES + AutoFlowS

The Operational View helps to understand the roles and responsibilities involved with resources being used within the INTERSECT Architecture. In Figure 3.1, we show a high-level diagram for the relevant resources that will be involved when connecting the AutoFlowS experimental environment at CNMS with computation resources at OLCF (e.g., ORNL Compute and Data Environment for Science (CADES) compute clusters). There are also some general resources like Data Transfer Node (DTN) and edge computing machines that will be used in a cross facility experiment that uses AutoFlowS at CNMS and Compute at OLCF.

The following outlines some of the Operational Steps associated with this high-level diagram. The intent is to highlight some of the anticipated steps to connect the resources and the Operational requirements necessary to facilitate these capabilities. CADES + AutoFlowS:

- Prepare facility accounts (CNMS, OLCF, possibly general ORNL).
- Setup any Network gateway/bridging services at CNMS (e.g., in/out network flows).
- Setup DTN Endpoints at CNMS with AutoFlowS data partition/storage area.
- Setup CADES software and/or VM images at OLCF (e.g., analysis software/codes).
- Setup DTN Endpoint and any new mounts at OLCF (e.g., Slate containers with parallel filesystems and DTN storage access).
- Setup Intersect Services (OLCF, CNMS, ORNL, ???).
- Verify bi-directional network flows for resources at OLCF and CNMS.
- Monitoring of DTN endpoints and certificate timeouts.
- Monitoring of data flows / performance.
- Monitoring of compute at OLCF (e.g., queue times for batch, load for dedicated, etc.).

[5]During design discussions, the *Command-Set* has also been referred to as a *Task*. However, the intent was primarily for grouping so the term *Command-Set* is being favored to promote this characteristic.

Roles:

- Network Administrators, System Administrators, Instrument Operators for the setup, configuration and monitoring of the resources used to operate the distributed system.
- Resource Owners will be responsible for approving accounts and allocations.

3.3 OPERATIONAL ACTIVITIES

The following subsections describe activities that are relevant from an operational view within the SoS architecture. The intent is to provide a general description, details for pertinent interfaces/data and include an illustrative example to clarify the activity.

Service Registry / Registrar

Service identity registration facility that provides the universally unique identifiers

Description

The INTERSECT services communicate via messages and require a unique address, i.e., UUID. The service registry provides a directory of systems with their respective services. These unique identifiers are used when communicating with targets. The current addressing structure follows a hierarchical format as shown in Figure 3.4.

Org	/	**Facility**	/	**System**	/	**Subsystem**	/	**Service**	/	**Capability**
[*name*]		[*name*]		[*name*|*uuid*]		[*name*|*uuid*]		[*uuid*]		[*name*]

Figure 3.4. Hierarchical naming structure for System of System addressing.

The registration service provides all entities in the INTERSECT ecosystem a unique identifier such that they may communicate on the messaging platform. The registry self-assigns UUIDs for the entities in the system. The active state of each service is not maintained in the registration beyond basic existence, i.e., the registry is not responsible for tracking service state (see Service Monitoring in Section 3.3). When a service registers it provides what capabilities (and version) it supports and receives a unique identifier for each capability, which get mapped down to the service schema. Additionally, a regular name may be included with services to aid with channel naming for broadcasts and be helpful for debugging.

Interfaces and Data

See Figure 3.5 for example of expected interfaces:

- `Register(Type,org,facility,[cert])` – connect to the Service Registry to be assigned a UUID, optional certificate for added security
- `Response(hash,UUID)` – response containing item's assigned UUID
- `RequestUUID(Type,org,facility,hash)` – request the UUID of a given item (hash ID), this may contain wildcards to obtain a set of UUID (e.g., "org.faciliity.*").

Example

The steps for a service to obtain a universally unique identifier are outlined in Figure 3.5.

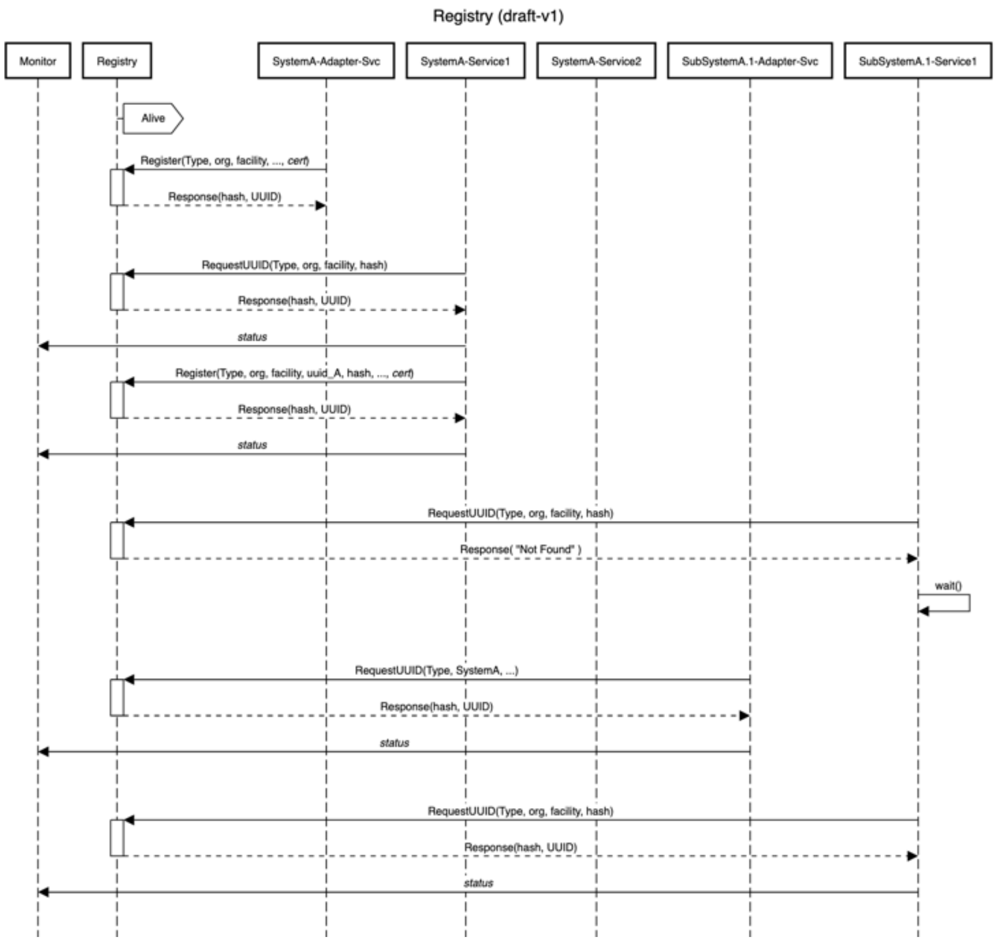


Figure 3.5. Illustration of messages involved in service registration to obtain a universally unique identifier (UUID).

Related

- Service Monitoring in Section 3.3

Service Monitoring

Observation facility to check the status (e.g., availability) of an INTERSECT service.

Description

The observation and reporting on the INTERSECT ecosystem will be a common operational procedure. This monitoring will focus on the INTERSECT Services and provide base mechanisms to gather high-level information about the distributed environment.

All Resources in INTERSECT will be exposed through some software Service; therefore, monitoring at the INTERSECT level should focus on the high-level state of these Services. The INTERSECT Monitoring Services can leverage existing facility/site monitoring capabilities but will normalize the information so it is generic across the distributed ecosystem. The monitoring will offer loose consistency, which will be easier to maintain if the information is at a high/summary level. There should be a well-defined set of states for the given service that enable INTERSECT users to reason about availability of the Service. Beyond the basic Operational/Availability status, the specific monitoring data for given Services will differ (i.e., a versioned schema for available monitoring data must be defined).

Interfaces and Data

The Monitoring Services should support the following commands:

- Connect – Join the monitoring system
- Disconnect – Gracefully leave the monitoring system
- Remove – Forcefully remove entity from monitoring system (e.g., unresponsive or dead Service)
- Status – "liveness" information (i.e., heartbeat) used to check if a Service is alive. Should be set to a given interval and include a triggering threshold. Upon failure to respond, state is set to "Unresponsive"", when exceed trigger threshold state set to "Dead".
- Info – show current state of monitoring system (e.g., connected resources/services), possibly support additional query flags to gather less/more detailed information. At minimum, should return identifiers for Services known at time within Monitoring system.
- (Optional) Health – customizable probe to gather Service specific details beyond simple liveness, possibly expensive to gather so care must be taken to avoid blocking and to avoid excessive load.

Monitoring States (see 'Status' command)

- Alive (responsive and functioning properly)
- Unresponsive (include timestamp for last 'heartbeat' message)
- Dead (failed state based on being unresponsive for longer than defined threshold period)
- Unknown (initialization state)

The monitoring service should support both a "push" and "pull" model of interaction. In the "pull" model, the user of monitoring data explicitly polls the service to get status. In contrast, the "push" model allows the user to register (subscribe) for asynchronous monitoring data updates sent out by the service. The details for this communication methods should be defined.

The monitoring service should honor INTERSECT authentication and security constraints (see also scoping/zones and Messaging on "visibility" private/public channels, etc.). The distribution of the monitoring data is also a point that will involve the Messaging and Data Plane, as will message ordering/timestamping.

Example

To illustrate the Service Monitor activity, we outline a hypothetical Compute Service Monitor, which includes a brief description of states and sequence diagrams to highlight interface patterns.

A compute service monitor might provide the following states:

- Operational States: Up / Down / Undefined
  - Up – compute service is available for the associated compute resource
  - Down – compute service is un-available for the associated compute resource
  - Undefined – compute service is in unknown state (possibly coming online/error/etc.)
- Availability States: Online / Offline / Undefined
  - Online – compute service is accepting requests
  - Offline – compute service is not accepting requests (e.g., maintenance mode)
  - Undefined – compute service is in unknown state (possibly coming online/error/etc.)

The sequence diagrams in Figure 3.6 illustrate steps for a *Compute Service Monitor*, possibly implemented via a “Sidecar Pattern” whereby Monitoring is coupled with the Compute Service to provide Operational/Availability “status” information. In Figure 3.6a, a User or Service queries for the monitoring status directly of an individual compute resource. In contrast, Figure 3.6b shows a more generic interface where the Compute Monitoring Service hides the available resources and monitoring requests about specific resources are only done within the Compute Monitor with external consumers just asking about generic status information. This Compute Monitoring Service acts as a clearing house for all the compute monitoring requests. In both cases, the INTERSECT Service hides the details of the backing source for the actual compute resource information and normalizes to the generic INTERSECT Monitoring schema. For example, *ComputeSvcMon* is the INTERSECT Service that talks to the opaque/internal *ComputeSchedulers A* and *B* to get details about the compute resources.

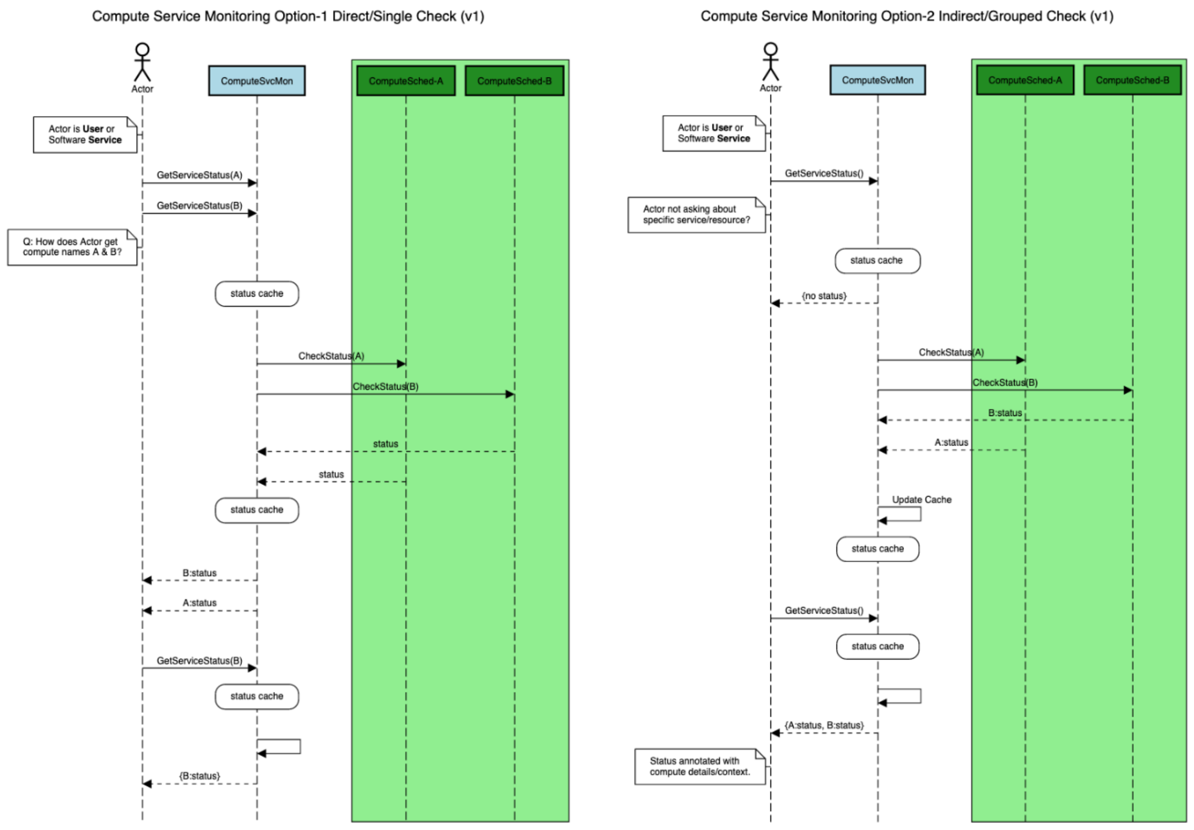
In Figure 3.7, we extrapolate the indirect/grouped scenario to an even more generic INTERSECT Monitoring Service (i.e., centralized monitoring service). This could be helpful for maintaining trends and performing analysis, that would be coupled with the high-level monitoring (data). This monitoring aggregator service can consume data from various INTERSECT Services and provide a general high-level view, or more global view of the broader system. The key is that the data for this is stored within this global monitoring service and not a requirement for the individual services to maintain historical data.

Advice to Implementors: The system should support caching of monitoring data with timeouts to avoid overwhelming services with monitoring requests.

Figure 3.8 recasts the *Compute Service Monitor* illustration in Figure 3.7 in terms of a “push” model, where the status updates are obtained via subscriptions. The *ComputeSvcMon* internally still uses a polling model in the diagram to gather state about the underlying resource.

Related

- Messaging
- Health and Diagnostics in Section 3.3



(a) Direct/Single check **(b) Indirect/Grouped check**

Figure 3.6. Sequence diagrams for illustration of Compute Service Monitoring using two possible approaches. Note, Blue box is INTERSECT Service, Green box/region is resource/opaque system.

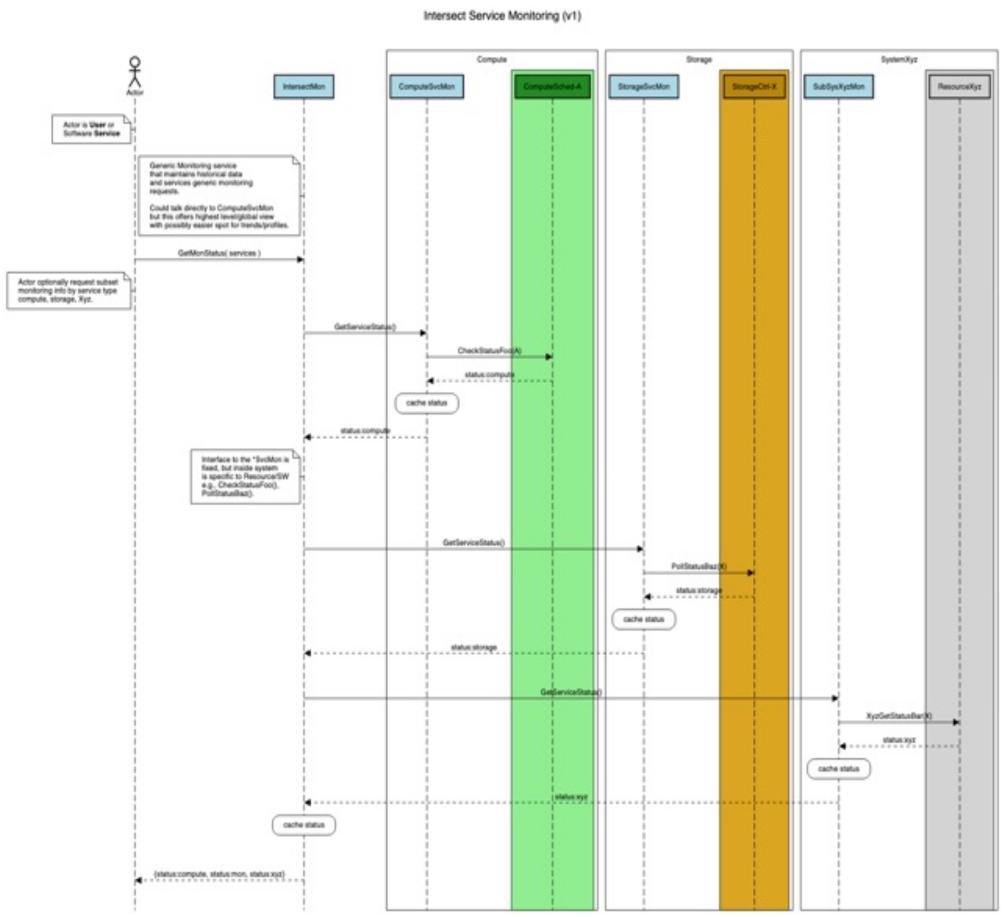

Intersect Service Monitoring (v1)
Compute
Storage
SystemXyz
Actor
IntersectMon
ComputeSvcMon
ComputeSched-A
StorageSvcMon
StorageCtrl-X
SubSysXyzMon
ResourceXyz
Actor is User or Software Service
Generic Monitoring service that maintains historical data and services generic monitoring requests.
Could talk directly to ComputeSvcMon but this offers highest level/global view with possibly easier spot for trends/profiles.
GetMonStatus(services)
Actor optionally request subset monitoring info by service type compute, storage, Xyz.
GetServiceStatus()
CheckStatusFoo(A)
status:compute
cache status
status:compute
Interface to the *SvcMon is fixed, but inside system is specific to Resource/SW e.g., CheckStatusFoo(), PollStatusBaz().
GetServiceStatus()
PollStatusBaz(X)
status:storage
cache status
status:storage
GetServiceStatus()
XyzGetStatusBar(X)
status:xyz
cache status
status:xyz
cache status
{status:compute, status:mon, status:xyz}

Figure 3.7. INTERSECT Service Monitor illustrating sequence diagram for interfacing with centralized Monitoring Service that stockpiles data about the various INTERSECT Services (e.g., Compute, Storage, etc.). Note, Blue box is INTERSECT Service, Green/Yellow/Grey boxes/regions are resources/opaque systems.

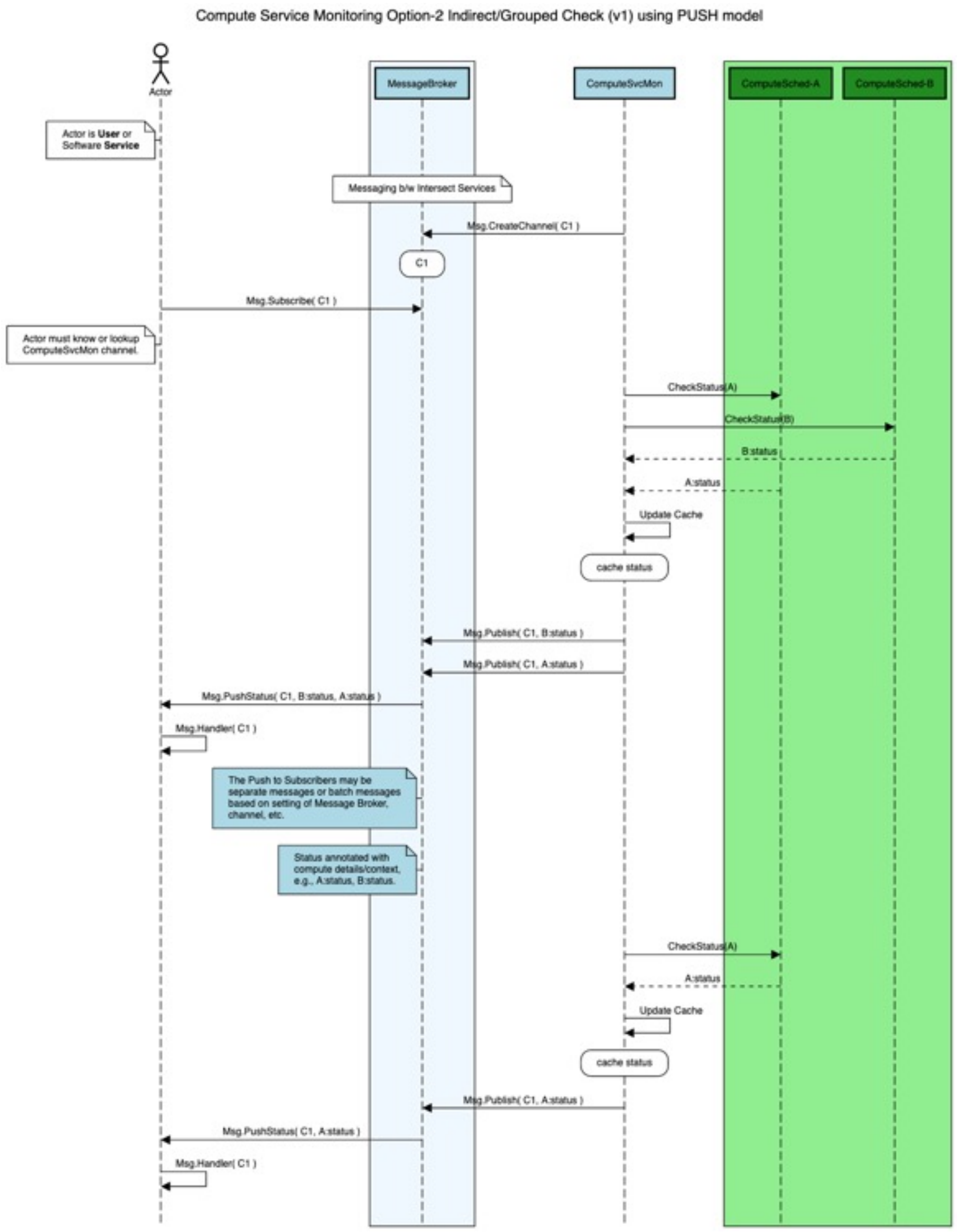

Compute Service Monitoring Option-2 Indirect/Grouped Check (v1) using PUSH model
Actor
MessageBroker
ComputeSvcMon
ComputeSched-A
ComputeSched-B
Actor is User or Software Service
Messaging b/w Intersect Services
Msg.CreateChannel(C1)
C1
Msg.Subscribe(C1)
Actor must know or lookup ComputeSvcMon channel.
CheckStatus(A)
CheckStatus(B)
B:status
A:status
Update Cache
cache status
Msg.Publish(C1, B:status)
Msg.Publish(C1, A:status)
Msg.PushStatus(C1, B:status, A:status)
Msg.Handler(C1)
The Push to Subscribers may be separate messages or batch messages based on setting of Message Broker, channel, etc.
Status annotated with compute details/context, e.g., A:status, B:status.
CheckStatus(A)
A:status
Update Cache
cache status
Msg.Publish(C1, A:status)
Msg.PushStatus(C1, A:status)
Msg.Handler(C1)

Figure 3.8. Illustration of "Push" model form of Computer Service Monitoring where Actor is updated asynchronously about monitoring status updates, and the ComputerSvcMon polls resources periodically about status and publishes to Message Broker that disseminates the messages to subscribers.

Health and Diagnostics

System health monitoring facility with probes to aid diagnostics and performance checks

Description

The distributed environment will need support for checking the overall health of systems and between systems. These services may require contextual details to help reason about the result of data gathered from other services (e.g., Service Monitoring, Data Movers, Schedulers). There should be support for registering probes that gather data, possibly on demand or via periodic timers, to help assess the health of the system. This probe and associated interpreter of the result are often coupled to make intelligent use of the data gathered. This service will require some state management to infer current "health" based on prior measurements. There should also be ways to actively query for the current status of items to determine their health at a given instance.

It may be useful to have support for aggregated data for time periods, e.g., round-robin database to have fixed size health monitoring at different granularities for historical time periods. Additionally, it will be useful to support methods to add "Probes" that can be inserted to dynamically gather data and to tailor tracking/diagnostic methods.

Advice to Implementors: The probes or extensible data processing methods might be additional micro-services to tailor the data for a given use case. This could lead to data coupling and restrictions on generality, while offering more advanced capabilities to perform diagnostics or other operations using information about INTERSECT resources.

Advice to Implementors: The Health and Diagnostic of Services may require specific input on what data is required to satisfy distributed diagnostic/profiling needs. This can be piggybacked on things like reporting of transfer times on file moves to gather performance data indirectly, or via periodically scheduled measurements between select endpoints in the federation. While related, these capabilities are more focused on the profiling and analytics and less about the generic monitoring of services.

Approval Processes

The INTERSECT services that manage resources will have access controls for Users/Projects that will includes either static or dynamic approvals.

Description

The addition of resources to an INTERSECT environment will require some form of approval. Additionally, the access controls will be managed by the Owner/Maintainer of the given resource. The access control system should allow for the granting and denial of a given User for a specific Resource, e.g., grant exclusive access for *userA* to an *instrumentX*. Note, the access to Resources is actually managed via INTERSECT services so the approval process is technically managing the associated Services for the Resource.

The approval process should support granting/denying access to the Resource on a per-User or per-Project basis. The approvals may be statically defined, e.g., configuration file, or dynamically managed via messages. The length of access, *access duration*, should be stated or indefinite if unbounded. The duration may be specified as a fixed length (i.e., ends at specific time/date) or as a relative duration from the start of when access is granted (i.e., ends one day after start). Once access is approved (granted), there should also be support for revoking the access that will trigger an interruption of service to the given User(s)/Project(s). Depending on the underlying Resource, this revocation may be delayed for some period of time until access removal is "safe", i.e., no future access but current operations may complete. The precise details for revocation delay can be defined on a per-service/resource basis.

Advice to Implementors: The INTERSECT approval process may assume pre-existing Users and Projects that are setup by the facility hosting the resource. The approval may trigger a standard account request process, but that is outside the scope of concern for the INTERSECT Approval interfaces. Restated, INTERSECT should not try to duplicate existing facility approval procedures, it may interface with existing processes but generally the Users/Projects will already exist and the access controls will simply enable/disable the existing accounts/groups.

Interfaces and Data

In the context of Experiment Planning [6], initial discussions have identified the need for approval messages during the creation of an experiment plan with the Orchestrator services.

Example

The *ApprovalRequest* and corresponding *ApprovalRequestStatus* message types were outlined in the context of Experiment Planning as shown in Figure 3.9. This sequence diagram outlines the steps and messages for Orchestrator and Application/Resource Manager services, which need support for requesting approval at different stages of a workflow.

Related

- User Views: Owner management operations in Sections A.2, A.2

[6]Based on whiteboard 27-apr-2022 notes related to ExperimentPlanApproval.

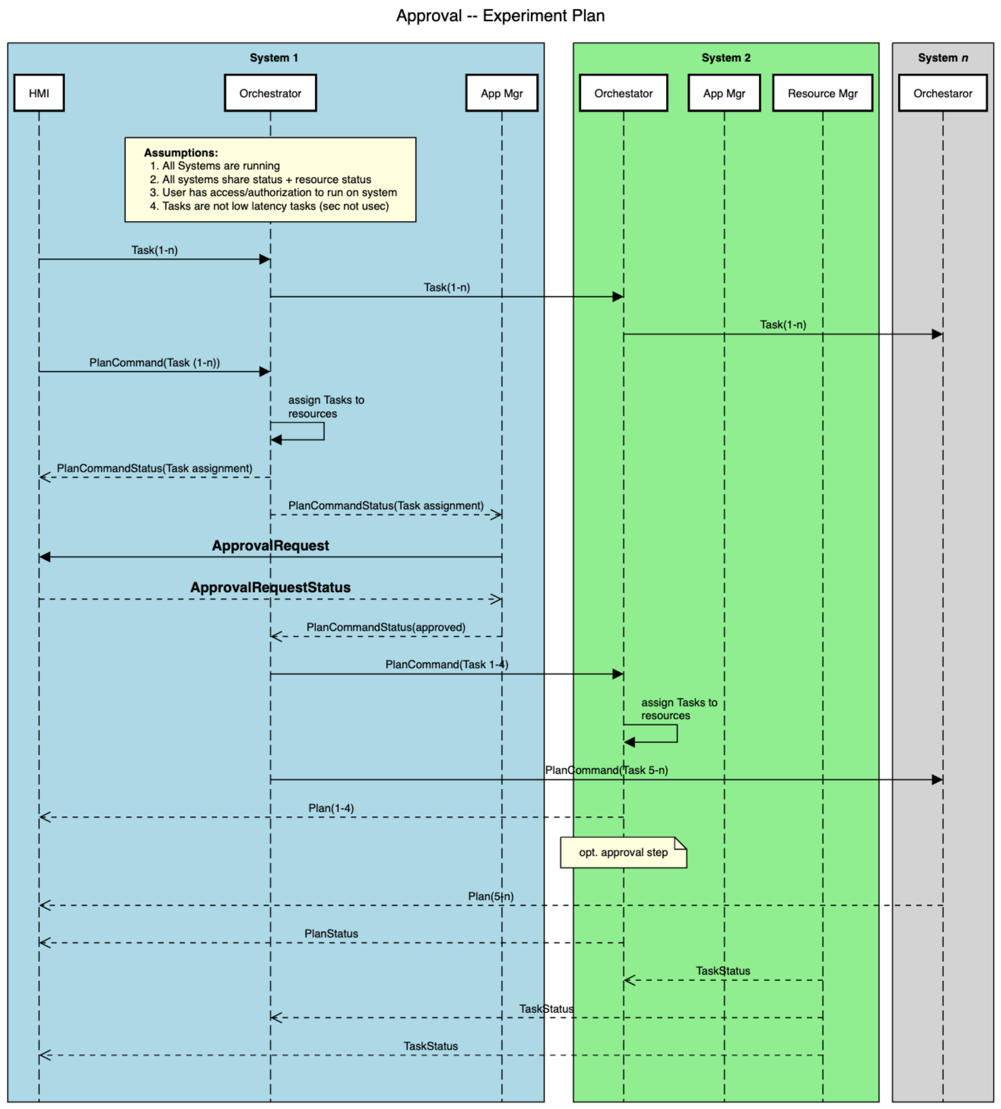

Approval -- Experiment Plan
System 1
HMI
Orchestrator
App Mgr
System 2
Orchestator
App Mgr
Resource Mgr
System *n*
Orchestaror
Assumptions:
1. All Systems are running
2. All systems share status + resource status
3. User has access/authorization to run on system
4. Tasks are not low latency tasks (sec not usec)
Task(1-n)
Task(1-n)
Task(1-n)
PlanCommand(Task (1-n))
assign Tasks to resources
PlanCommandStatus(Task assignment)
PlanCommandStatus(Task assignment)
ApprovalRequest
ApprovalRequestStatus
PlanCommandStatus(approved)
PlanCommand(Task 1-4)
assign Tasks to resources
PlanCommand(Task 5-n)
Plan(1-4)
opt. approval step
Plan(5-n)
PlanStatus
TaskStatus
TaskStatus
TaskStatus

Figure 3.9. Sequence diagram for experiment planning workflow that shows an ApprovalRequest/ApprovalRequestStatus message exchange.

4 USER VIEW

This section of the document will describe the User View, including the view's definition, user roles, what the view does and does not include, and detailed user view sections based on Person Type and activity.

4.1 INTRODUCTION

This section introduces the User View, including the definition, user roles, and what this view does and does not include.

User View Definition

The user view is a representation of a system of systems that illustrates different user's interactions with the INTERSECT system. It does not include interactions between INTERSECT components themselves. This view will serve to highlight the user facing functionality required from INTERSECT.

User Roles

The DoDAF meta-model defines concepts involving performer. The DoDAF definition of performer is any entity – human, automated, or any aggregation of human and/or automated – that performs an activity and provides a capability.

- **Organization**: A specific real-world assemblage of people and other resources organized for an on-going purpose.
- **System**: A functionally, physically, and/or behaviorally related group of regularly interacting or interdependent elements.
- **Person Type**: A category of persons defined by the role or roles they share that are relevant to an architecture.
- **Service**: A mechanism to enable access to a set of one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description. The mechanism is a Performer. The capabilities accessed are Resources – Information, Data, Materiel, Performers, and Geo-political Extents.

DoDAF defines *capability* as, "the ability to achieve a Desired Effect under specified (performance) standards and conditions through combinations of ways and means (activities and resources) to perform a set of activities."

DoDAF defines *activity* as work, not specific to a single organization, weapon system or individual that transforms inputs (Resources) into outputs (Resources) or changes their state.

Table 4.1 defines the Person Type and their role description as it applies to the INTERSECT architecture.

Note that user roles are contextual, typically limited to the scope of Resources. For example, an Owner of a Resource would have elevated privileges only for the Resources they own; they would be considered a regular User in the context of any other Resource they don't own.

What is in the user view?

Examples of the components in the user view are identified per user role, such as:

Person Type	Description	Example
User	Using the system (not responsible for administration)	A User leverages the SDK on multiple Resources to compose and run a scientific Campaign.
Maintainer / Operator	Maintains one or more resources; different view of system (i.e., in contrast to User).	An Operator installs and configures the SDK on one or more Resources in addition to setting up (e.g. - loading chemicals into vials) and tearing down (e.g. purging tubes of all chemicals) the setup for each Campaign
Administrator	Maintains one or more systems; complete view of "their"system (their jurisdiction/domain/realm/area). Limited to a given jurisdiction (i.e., their administrative domain).	An Administrator grants/approves new Resources, where the SDK is installed, to be added to the INTERSECT ecosystem
Owner	Individual fiscally responsible for a resource; Vested interest; Possibly approver for a resource	An Owner purchases Resource(s) and delegates maintenance and operations to Operators. Owners could be Group Leaders to an Associate Lab Director
Provider	Manufacturer of a Logical (e.g. INTERSECT services) or Physical Resource (Computing, Observational, Data, Networking). Provider creates the Resource that the Operator maintains.	Examples include manufacturer of a scientific instrument, NVIDIA for an edge compute Resource, INTERSECT SecDevOps is provider of core INTERSECT SDK

Table 4.1. Person Type roles and descriptions.

- User applying for an INTERSECT account.
- User logging into their INTERSECT account.
- User dashboard.
- User submitting campaigns to INTERSECT.
- Notifications.
- Owner dashboard.
- Maintainer/Operator dashboard.
- Administrator dashboard.

What is NOT in the user view?

Data flows, communication, and any interaction between INTERSECT components would not be part of this view. Any activity that is unseen by the user is not part of the user view.

- The user's / beamline scientist's / technician's physical interaction with an instrument for non-automatable tasks such as:
  - Configuring a Campaign including connecting electrical cables or measurement leads, plumbing tubes, turning on or off (vacuum) pumps, etc.
  - Unloading or loading a sample into or onto an instrument.

  - Adjusting instrument parameters such as changing the magnification on a microscope by swapping a lens.
  - Calibrating an instrument manually.
  - Resetting the instrument after and/or before Campaigns. This could even mean emptying out beakers / test tubes and cleaning out chemicals such as in the case of AutoFlows.

Appendix A describes the different views associated with the User person type as described in Table 4.1. Diagrams of user interfaces are intended to illustrate data and actions available to the User in particular views; they are informational and are not intended to represent actual graphical user interfaces (GUIs) design or implementation components.

5 DATA VIEW

This section of the architecture specification defines the Data View.

5.1 INTRODUCTION

Data View Definition

The data view of a system of systems is a representation of the system from the perspective of data needs, and the data framework that needs to exist to support the INTERSECT architecture. The data view is a specification for all data aspects of INTERSECT as a whole system, and shall include the conceptual, logical, and physical data models. The conceptual data model provides the high-level data concepts and their relationships that are important to INTERSECT's operations that meet its intended purpose. The logical data model bridges the conceptual and physical data models and introduces the data structure for needed components. The physical data model is the actual data schema and specifications for INTERSECT services and applications.

What is in the data view?

Some of the items associated with the data view are:

- Overall data flow within the INTERSECT system.
- Descriptions, definitions of data components.
- Specification of the data messages (e.g. XML schema) for command/control, data movement of bulk and streaming data.
- Database schema for INTERSECT's operational data.
- Sequence diagrams to show data exchange.

What is NOT in the data view?

Some of the items NOT associated with the data view are:

- Specifications for scientific data (aka, the payload).
- Specifications for instruments, resources, etc.
- Experiment specifications.

Stakeholder and concerns

This view should be read by software developers, database developers, software engineers, network engineers, project managers, and others who will need to understand the data.

5.2 ENTITY RELATIONSHIP DATA MODEL

This section describes the entity-relationship data model of the INTERSECT ecosystem, as depicted in Figure 5.1.

Entity Name	Description

Entity Name	Description
User	A user of an INTERSECT-compliant system or application. May participate in authentication or authorization processes.
User Profile	Profile information (contact/address/miscellaneous) for an INTERSECT user.
Project	Accounting abstraction for resource allocation in an INTERSECT system.
Campaign	A collection of related experimental activity which uses INTERSECT resources. A Campaign is associated with a Project and may have multiple Users associated with it. Campaigns have explicit durations and discrete sets of resources assigned to them.
Campaign Result	Outcomes of INTERSECT Campaigns. There may be several different result states represented.
Campaign Error	“Error” outcomes for INTERSECT Campaigns. As with Campaign Result, there may be several different “flavors” of error/failure results.
Campaign Template	It may prove useful to memoize a Campaign structure as a template, so that it may be quickly replicated by users. Such repllicated new Campaigns are assigned the tamplated INTERSECT resources.
Recipe	Users may also wish to reuse resource structures at a finer granularity than Campaign. Recipies allow this usage to be memoized.
Approved User Resources Approved Administrator Resources Approved Operator Resources	Resource allocations are tracked with approval durations for each of Users, Administrators, and Operators.
INTERSECT Resource Type	Additional information about an INTERSECT resource.
INTERSECT Resource Action	Detail on the operations/functions available from a given INTERSECT resource.
INTERSECT Resources	Experimental/physical, computational, or virtual facilities available within the INTERSECT system or application.
Computational Resource	Additional information about computational resources available to the INTERSECT system or application.
Resource Support	An INTERSECT resource may be large and complex, requiring specialized support procedures and/or personnel for operation. Computational resources, for example, may have multiple such support staff, organized into tiers or functional areas.
Resource Capability	Resources provide INTERSECT capabilities, which allow them to be composed into systems and applications within the INTERSECT Architecture.

Table 5.1. Names and descriptions of INTERSECT architecture data entities

Table 5.1 presents detail on the entities shown in Figure 5.1.

5.3 INTERSECT DATA MESSAGING SCHEMA

See Appendix B for the INTERSECT XML messaging schema, which is explained below.

As depicted in Figure 5.2, there are 5 base message types in the INTERSECT messaging schema:

- command
- acknowledge
- request
- reply
- event

The *CommandMessage* type is a command message sent from a source to a target that requires an action from the target. When the target receives the message, they must acknowledge receipt of the message immediately with an *AcknowledgeMessage* type. Processing of the actual command commences any time after acknowledgement.

The *AcknowledgeMessage* type is a response to a command message. The response is either “ACCEPTED” or “REJECTED”, with any optional details provided in the message body. This message only indicates if a *CommandMessage* has been accepted for processing and not the status of the processing.

The *RequestMessage* type is a request message sent from a source to a target that is more inquisitive and requires a response from the target. When the target receives the message, the target deciphers the request found in the message body, processes the request, and then sends a reply to the source using the *ReplyMessage* type.

The *ReplyMessage* type is a response to a request message. The response is found in the message body and provides detailed information.

The *EventMessage* type is a notification message type that is either directed at a single target or multiple targets (broadcast). This message is asynchronous and does not need a response or acknowledgement. Some examples of *EventMessage* are heartbeat messages, log messages, and status to completion messages.

Each *ScienceEcosystemMessage* will contain only one of these 5 base message types per message. Each message has a similar “header” and the body content is an XML instance based on the schema provided in the header. This will allow the system to expand and incorporate many different components without the need for a single, monolithic schema.

The details of each element, attribute, and type, along with the URIs of any inherited types, are described below.

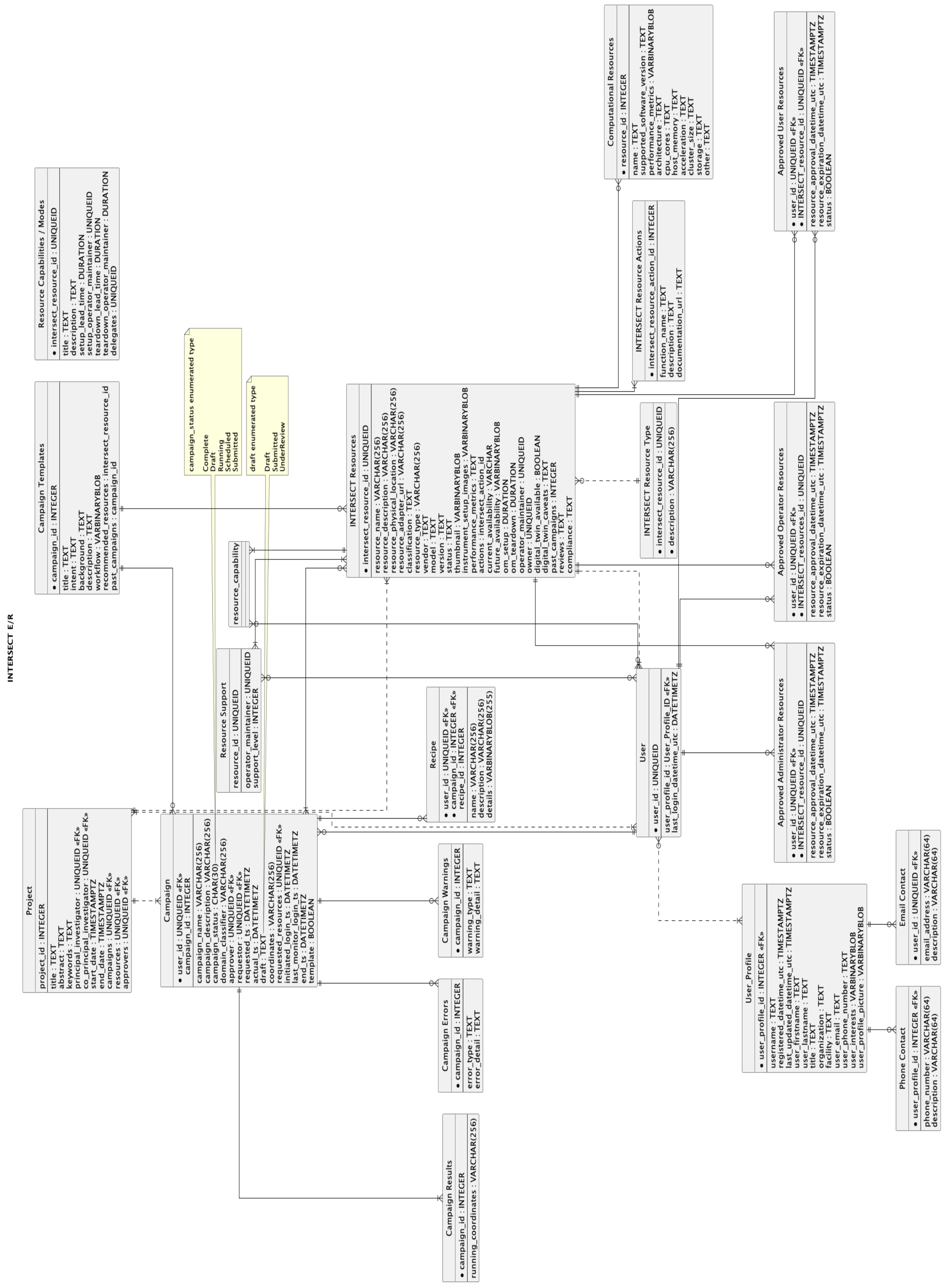

INTERSECT E/R
Resource Capabilities / Modes
● intersect_resource_id : UNIQUEID
title : TEXT
description : TEXT
setup_lead_time : DURATION
setup_operator_maintainer : UNIQUEID
teardown_lead_time : DURATION
teardown_operator_maintainer : DURATION
delegates : UNIQUEID
Campaign Templates
● campaign_id : INTEGER
title : TEXT
intent : TEXT
background : TEXT
description : TEXT
workflow : VARBINARYBLOB
recommended_resources : intersect_resource_id
past_campaigns : campaign_id
campaign_status enumerated type
Complete
Draft
Running
Scheduled
Submitted
draft enumerated type
Draft
Submitted
UnderReview
Computational Resources
● resource_id : INTEGER
name : TEXT
supported_software_version : TEXT
performance_metrics : VARBINARYBLOB
architecture : TEXT
cpu_cores : TEXT
host_memory : TEXT
acceleration : TEXT
cluster_size : TEXT
storage : TEXT
other : TEXT
INTERSECT Resource Actions
● intersect_resource_action_id : INTEGER
function_name : TEXT
description : TEXT
documentation_url : TEXT
Approved User Resources
● user_id : UNIQUEID «FK»
● INTERSECT_resource_id : UNIQUEID «FK»
resource_approval_datetime_utc : TIMESTAMPTZ
resource_expiration_datetime_utc : TIMESTAMPTZ
status : BOOLEAN
INTERSECT Resources
● intersect_resource_id : UNIQUEID
resource_name : VARCHAR(256)
resource_description : VARCHAR(256)
resource_physical_location : VARCHAR(256)
resource_adapter_url : VARCHAR(256)
classification : TEXT
resource_type : VARCHAR(256)
vendor : TEXT
model : TEXT
version : TEXT
status : TEXT
thumbnail : VARBINARYBLOB
instrument_setup_images : VARBINARYBLOB
performance_metrics : TEXT
actions : intersect_action_id
current_availability : VARCHAR
future_availability : VARBINARYBLOB
om_setup : DURATION
om_teardown : DURATION
operator_maintainer : UNIQUEID
owner : UNIQUEID
digital_twin_available : BOOLEAN
digital_twin_caveats : TEXT
past_campaigns : INTEGER
reviews : TEXT
compliance : TEXT
INTERSECT Resource Type
● intersect_resource_id : UNIQUEID
● description : VARCHAR(256)
Approved Operator Resources
● user_id : UNIQUEID «FK»
● INTERSECT_resources_id : UNIQUEID
resource_approval_datetime_utc : TIMESTAMPTZ
resource_expiration_datetime_utc : TIMESTAMPTZ
status : BOOLEAN
resource_capability
Resource Support
resource_id : UNIQUEID
operator_maintainer : UNIQUEID
support_level : INTEGER
Recipe
● user_id : UNIQUEID «FK»
● campaign_id : INTEGER «FK»
● recipe_id : INTEGER
name : VARCHAR(256)
description : VARCHAR(256)
details : VARBINARYBLOB(255)
User
● user_id : UNIQUEID
user_profile_id : User_Profile_ID «FK»
last_login_datetime_utc : DATETIMETZ
Approved Administrator Resources
● user_id : UNIQUEID «FK»
● INTERSECT_resource_id : UNIQUEID
resource_approval_datetime_utc : TIMESTAMPTZ
resource_expiration_datetime_utc : TIMESTAMPTZ
status : BOOLEAN
Project
project_id : INTEGER
title : TEXT
abstract : TEXT
keywords : TEXT
principal_investigator : UNIQUEID «FK»
co_principal_investigator : UNIQUEID «FK»
start_date : TIMESTAMPTZ
end_date : TIMESTAMPTZ
campaigns : UNIQUEID «FK»
resources : UNIQUEID «FK»
approvers : UNIQUEID «FK»
Campaign
● user_id : UNIQUEID «FK»
campaign_id : INTEGER
campaign_name : VARCHAR(256)
campaign_description : VARCHAR(256)
campaign_status : CHAR(30)
domain_classifier : VARCHAR(256)
approver : UNIQUEID «FK»
requestor : UNIQUEID «FK»
requested_ts : DATETIMETZ
actual_ts : DATETIMETZ
draft : TEXT
coordinates : VARCHAR(256)
requested_resources : UNIQUEID «FK»
initiated_login_ts : DATETIMETZ
last_monitor_login_ts : DATETIMETZ
end_ts : DATETIMETZ
template : BOOLEAN
Campaign Warnings
● campaign_id : INTEGER
warning_type : TEXT
warning_detail : TEXT
Campaign Errors
● campaign_id : INTEGER
error_type : TEXT
error_detail : TEXT
Campaign Results
● campaign_id : INTEGER
running_coordinates : VARCHAR(256)
User_Profile
● user_profile_id : INTEGER «FK»
username : TEXT
registered_datetime_utc : TIMESTAMPTZ
last_updated_datetime_utc : TIMESTAMPTZ
user_firstname : TEXT
user_lastname : TEXT
title : TEXT
organization : TEXT
facility : TEXT
user_email : TEXT
user_phone_number : TEXT
user_interests : VARBINARYBLOB
user_profile_picture : VARBINARYBLOB
Email Contact
● user_id : UNIQUEID «FK»
email_address : VARCHAR(64)
description : VARCHAR(64)
Phone Contact
● user_profile_id : INTEGER «FK»
phone_number : VARCHAR(64)
description : VARCHAR(64)

Figure 5.1. Entity-Relationship Diagram for INTERSECT.

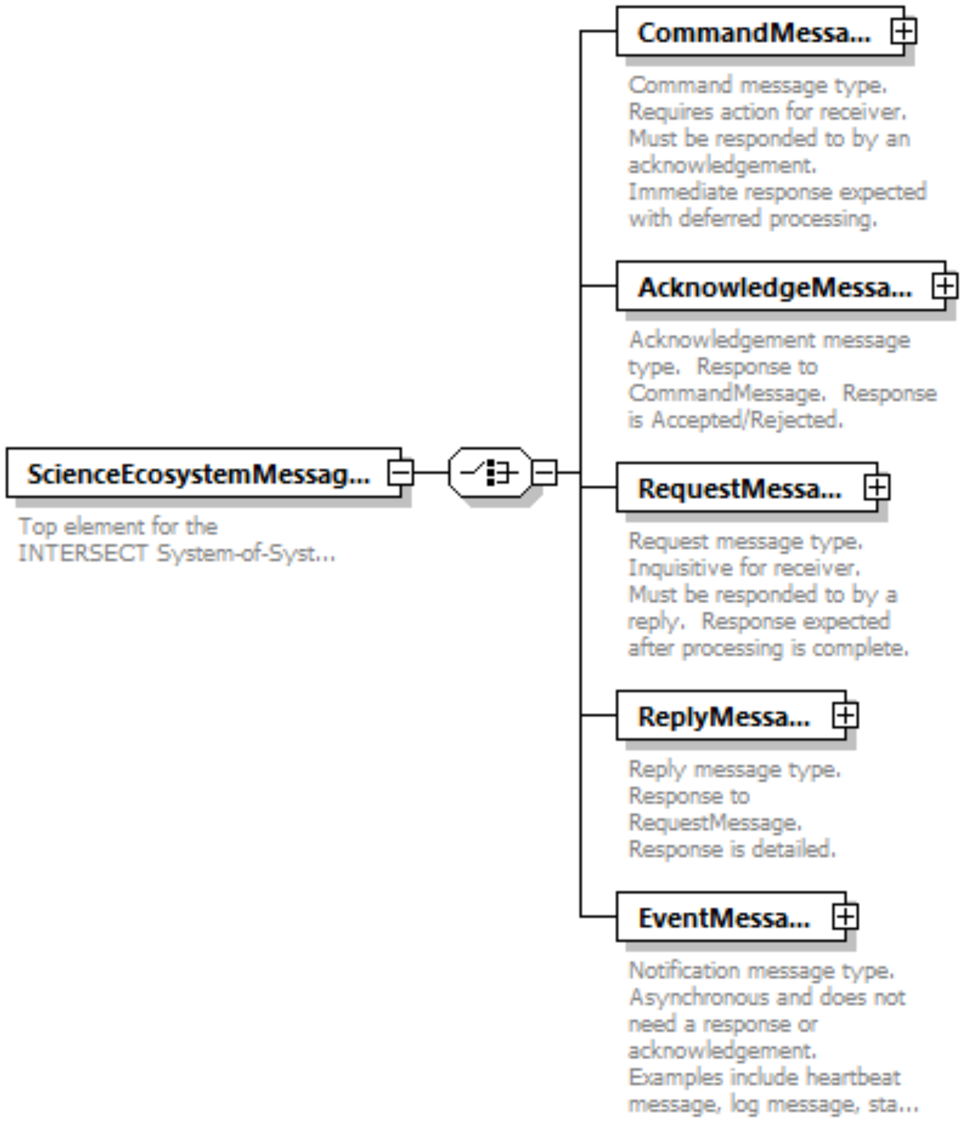

ScienceEcosystemMessag...
Top element for the INTERSECT System-of-Syst...
CommandMessa...
Command message type. Requires action for receiver. Must be responded to by an acknowledgement. Immediate response expected with deferred processing.
AcknowledgeMessa...
Acknowledgement message type. Response to CommandMessage. Response is Accepted/Rejected.
RequestMessa...
Request message type. Inquisitive for receiver. Must be responded to by a reply. Response expected after processing is complete.
ReplyMessa...
Reply message type. Response to RequestMessage. Response is detailed.
EventMessa...
Notification message type. Asynchronous and does not need a response or acknowledgement. Examples include heartbeat message, log message, sta...

Figure 5.2. Top level of the INTERSECT XML schema.

Schema Document Properties

Target Namespace	None
Element and Attribute Namespaces	• Global element and attribute declarations belong to this schema's target namespace. • By default, local element declarations belong to this schema's target namespace. • By default, local attribute declarations have no namespace.

Declared Namespaces

Prefix	Namespace
xml	http://www.w3.org/XML/1998/namespace
vc	http://www.w3.org/2007/XMLSchema-versioning
xs	http://www.w3.org/2001/XMLSchema

Global Declarations

Element: MessageContent

Name	MessageContent
Type	xs:base64Binary
Nillable	no
Abstract	no
Documentation	The actual message content, deciphered by either MessageSchemaURI or (MessageType, MessageName, MessageVersion).

XML Instance Representation

```
<MessageContent>xs:base64Binary</MessageContent>
```

Schema Component Representation

```
<xs:element name="MessageContent" type="xs:base64Binary" />
```

Element: MessageCreationDateTimeUTC

Name	MessageCreationDateTimeUTC
Type	xs:dateTime
Nillable	no
Abstract	no

Documentation	The UTC date and time that this message instance was created.

XML Instance Representation

```
<MessageCreationDateTimeUTC>xs:dateTime</MessageCreationDateTimeUTC>
```

Schema Component Representation

```
<xs:element name="MessageCreationDateTimeUTC" type="xs:dateTime" />
```

Element: MessageID

Name	MessageID
Type	xs:ID
Nillable	no
Abstract	no
Documentation	The ID of this message. If "Command" or "Request", this is the ID used in the "Ack" or "Reply" message.

XML Instance Representation

```
<MessageID>xs:ID</MessageID>
```

Schema Component Representation

```
<xs:element name="MessageID" type="xs:ID" />
```

Element: MessageName

Name	MessageName
Type	Locally-defined simple type
Nillable	no
Abstract	no
Documentation	The name for the given message type.

XML Instance Representation

```
<MessageName>xs:string (length >= 1) </MessageName>
```

Schema Component Representation

```
<xs:element name="MessageName" >
  <xs:simpleType>
  <xs:restriction base="xs:string" >
  <xs:minLength value="1" />
```

```
</xs:restriction>
</xs:simpleType>
</xs:element>
```

Element: MessageSchemaURI

Name	MessageSchemaURI
Type	xs:anyURI
Nillable	no
Abstract	no
Documentation	The URL of the schema to decipher the MessageContent element.

XML Instance Representation

```
<MessageSchemaURI>xs:anyURI</MessageSchemaURI>
```

Schema Component Representation

```
<xs:element name="MessageSchemaURI" type="xs:anyURI" />
```

Element: MessageType

Name	MessageType
Type	xs:string
Nillable	no
Abstract	no
Documentation	The type of message - enumerated list. See ScienceEcosystemMessageTypes.xsd for the current listing.

XML Instance Representation

```
<MessageType>xs:string</MessageType>
```

Schema Component Representation

```
<xs:element name="MessageType" type="xs:string" />
```

Element: MessageVersion

Name	MessageVersion
Type	Locally-defined simple type
Nillable	no
Abstract	no
Documentation	The version of the message type being used in the MessageContent element.

XML Instance Representation

```
<MessageVersion>xs:decimal (value >= 0)</MessageVersion>
```

Schema Component Representation

```
<xs:element name="MessageVersion" >
  <xs:simpleType>
  <xs:restriction base="xs:decimal" >
  <xs:minInclusive value="0" />
</xs:restriction>
</xs:simpleType>
</xs:element>
```

Element: ResponseSummary

Name	ResponseSummary
Type	Locally-defined simple type
Nillable	no
Abstract	no
Documentation	Enumerated type of SUCCESS or FAILURE only. More information is provided in MessageContent element.

XML Instance Representation

```
<ResponseSummary>xs:string (value comes from list: {'SUCCESS'|'FAILURE'})</ResponseSummary>
```

Schema Component Representation

```
<xs:element name="ResponseSummary" >
  <xs:simpleType>
  <xs:restriction base="xs:string" >
  <xs:enumeration value="SUCCESS" />
  <xs:enumeration value="FAILURE" />
</xs:restriction>
</xs:simpleType>
</xs:element>
```

Element: ResponseToMessageID

Name	ResponseToMessageID
Type	xs:ID
Nillable	no
Abstract	no
Documentation	The MessageID relating to the original request.

XML Instance Representation

```
<ResponseToMessageID>xs:ID</ResponseToMessageID>
```

Schema Component Representation

```
<xs:element name="ResponseToMessageID" type="xs:ID" />
```

Element: ScienceEcosystemMessages

Name	ScienceEcosystemMessages
Type	Locally-defined complex type
Nillable	no
Abstract	no
Documentation	Top element for the INTERSECT System-of-Systems Messages.

XML Instance Representation

```
<ScienceEcosystemMessages>
Start Choice[1]
    <CommandMessage>CommandMessageType</CommandMessage>[1]
    <AcknowledgeMessage>AcknowledgeMessageType</AcknowledgeMessage>[1]
    <RequestMessage>RequestMessageType</RequestMessage>[1]
    <ReplyMessage>ReplyMessageType</ReplyMessage>[1]
    <EventMessage>EventMessageType</EventMessage>[1]
End Choice
</ScienceEcosystemMessages>
```

Schema Component Representation

```
<xs:element name="ScienceEcosystemMessages" >
  <xs:complexType>
  <xs:choice>
  <xs:element name="CommandMessage" type="CommandMessageType" />
  <xs:element name="AcknowledgeMessage" type="AcknowledgeMessageType" />
  <xs:element name="RequestMessage" type="RequestMessageType" />
  <xs:element name="ReplyMessage" type="ReplyMessageType" />
  <xs:element name="EventMessage" type="EventMessageType" />
</xs:choice>
</xs:complexType>
</xs:element>
```

Element: SourceID

Name	SourceID

Type	anyType
Nillable	no
Abstract	no
Documentation	The sender ID of this message. This is the ID of the component or system at which this message was formulated.

XML Instance Representation

```
<SourceID>...</SourceID>
```

Schema Component Representation

```
<xs:element name="SourceID" />
```

Element: TargetID

Name	TargetID
Type	anyType
Nillable	no
Abstract	no
Documentation	The receiver ID of this message. This is the ID of the component or system at which this message was sent.

XML Instance Representation

```
<TargetID>...</TargetID>
```

Schema Component Representation

```
<xs:element name="TargetID" />
```

Global Definitions

Complex Type: AcknowledgeMessageType

Super-types:	None
Sub-types:	None

Name	AcknowledgeMessageType
Abstract	no

XML Instance Representation

```
<...>
  <MessageID>...</MessageID>[1]
  <MessageCreationDateTimeUTC>...</MessageCreationDateTimeUTC>[1]
  <SourceID>...</SourceID>[1]
  <TargetID>...</TargetID>[1]
  <ResponseToMessageID>...</ResponseToMessageID>[1]
  <ResponseSummary>...</ResponseSummary>[1]
  <MessageSchemaURI>...</MessageSchemaURI>[1]
  <MessageType>...</MessageType>[0..1]
  <MessageName>...</MessageName>[0..1]
  <MessageVersion>...</MessageVersion>[0..1]
  <MessageContent>...</MessageContent>[1]
</...>
```

Schema Component Representation

```
<xs:complexType name="AcknowledgeMessageType" >
  <xs:sequence>
  <xs:element ref="MessageID" />
  <xs:element ref="MessageCreationDateTimeUTC" />
  <xs:element ref="SourceID" />
  <xs:element ref="TargetID" />
  <xs:element ref="ResponseToMessageID" />
  <xs:element ref="ResponseSummary" />
  <xs:element ref="MessageSchemaURI" />
  <xs:element ref="MessageType" minOccurs="0" />
  <xs:element ref="MessageName" minOccurs="0" />
  <xs:element ref="MessageVersion" minOccurs="0" />
  <xs:element ref="MessageContent" />
</xs:sequence>
</xs:complexType>
```

Complex Type: CommandMessageType

Super-types:	None
Sub-types:	None

Name	CommandMessageType
Abstract	no
Documentation	Base type for a CommandMessage.

XML Instance Representation

```
<...>
  <MessageID>...</MessageID>[1]
  <MessageCreationDateTimeUTC>...</MessageCreationDateTimeUTC>[1]
  <SourceID>...</SourceID>[1]
  <TargetID>...</TargetID>[1]
  <MessageSchemaURI>...</MessageSchemaURI>[1]
  <MessageType>...</MessageType>[0..1]
  <MessageName>...</MessageName>[0..1]
  <MessageVersion>...</MessageVersion>[0..1]
  <MessageContent>...</MessageContent>[1]
</...>
```

Schema Component Representation

```
<xs:complexType name="CommandMessageType" >
  <xs:sequence>
  <xs:element ref="MessageID" />
  <xs:element ref="MessageCreationDateTimeUTC" />
  <xs:element ref="SourceID" />
  <xs:element ref="TargetID" />
  <xs:element ref="MessageSchemaURI" />
  <xs:element ref="MessageType" minOccurs="0" />
  <xs:element ref="MessageName" minOccurs="0" />
  <xs:element ref="MessageVersion" minOccurs="0" />
  <xs:element ref="MessageContent" />
</xs:sequence>
</xs:complexType>
```

Complex Type: EventMessageType

Super-types:	None
Sub-types:	None

Name	EventMessageType
Abstract	no
Documentation	Base type for a EventMessage.

XML Instance Representation

```
<...>
  <MessageID>...</MessageID>[1]
  <MessageCreationDateTimeUTC>...</MessageCreationDateTimeUTC>[1]
  <SourceID>...</SourceID>[1]
  <TargetID>...</TargetID>[1]
  <MessageSchemaURI>...</MessageSchemaURI>[1]
  <MessageType>...</MessageType>[0..1]
  <MessageName>...</MessageName>[0..1]
  <MessageVersion>...</MessageVersion>[0..1]
  <MessageContent>...</MessageContent>[1]
</...>
```

Schema Component Representation

```
<xs:complexType name="EventMessageType" >
  <xs:sequence>
  <xs:element ref="MessageID" />
  <xs:element ref="MessageCreationDateTimeUTC" />
  <xs:element ref="SourceID" />
  <xs:element ref="TargetID" />
  <xs:element ref="MessageSchemaURI" />
  <xs:element ref="MessageType" minOccurs="0" />
  <xs:element ref="MessageName" minOccurs="0" />
  <xs:element ref="MessageVersion" minOccurs="0" />
  <xs:element ref="MessageContent" />
</xs:sequence>
</xs:complexType>
```

Complex Type: ReplyMessageType

Super-types:	None
Sub-types:	None

Name	ReplyMessageType
Abstract	no
Documentation	Base type for a ReplyMessage.

XML Instance Representation

```
<...>
  <MessageID>...</MessageID>[1]
  <MessageCreationDateTimeUTC>...</MessageCreationDateTimeUTC>[1]
  <SourceID>...</SourceID>[1]
  <TargetID>...</TargetID>[1]
  <ResponseToMessageID>...</ResponseToMessageID>[1]
  <ResponseSummary>...</ResponseSummary>[1]
  <MessageSchemaURI>...</MessageSchemaURI>[1]
  <MessageType>...</MessageType>[0..1]
  <MessageName>...</MessageName>[0..1]
  <MessageVersion>...</MessageVersion>[0..1]
  <MessageContent>...</MessageContent>[1]
</...>
```

Schema Component Representation

```
<xs:complexType name="ReplyMessageType" >
  <xs:sequence>
  <xs:element ref="MessageID" />
  <xs:element ref="MessageCreationDateTimeUTC" />
  <xs:element ref="SourceID" />
  <xs:element ref="TargetID" />
  <xs:element ref="ResponseToMessageID" />
  <xs:element ref="ResponseSummary" />
  <xs:element ref="MessageSchemaURI" />
  <xs:element ref="MessageType" minOccurs="0" />
  <xs:element ref="MessageName" minOccurs="0" />
  <xs:element ref="MessageVersion" minOccurs="0" />
  <xs:element ref="MessageContent" />
</xs:sequence>
</xs:complexType>
```

Complex Type: RequestMessageType

Super-types:	None
Sub-types:	None

Name	RequestMessageType
Abstract	no
Documentation	Base type for a RequestMessage.

XML Instance Representation

```
<...>
  <MessageID>...</MessageID>[1]
  <MessageCreationDateTimeUTC>...</MessageCreationDateTimeUTC>[1]
  <SourceID>...</SourceID>[1]
  <TargetID>...</TargetID>[1]
  <MessageSchemaURI>...</MessageSchemaURI>[1]
  <MessageType>...</MessageType>[0..1]
  <MessageName>...</MessageName>[0..1]
  <MessageVersion>...</MessageVersion>[0..1]
  <MessageContent>...</MessageContent>[1]
```

```
</...>
```

Schema Component Representation

```
<xs:complexType name="RequestMessageType" >
  <xs:sequence>
  <xs:element ref="MessageID" />
  <xs:element ref="MessageCreationDateTimeUTC" />
  <xs:element ref="SourceID" />
  <xs:element ref="TargetID" />
  <xs:element ref="MessageSchemaURI" />
  <xs:element ref="MessageType" minOccurs="0" />
  <xs:element ref="MessageName" minOccurs="0" />
  <xs:element ref="MessageVersion" minOccurs="0" />
  <xs:element ref="MessageContent" />
</xs:sequence>
</xs:complexType>
```

5.4 EXAMPLE MESSAGING SEQUENCE DIAGRAMS

Component Registration

Sequence diagram to depict messages for registering an INTERSECT component is shown in Figure 5.3.

User Login

Sequence diagram to depict messages for user login is shown in Figure 5.4.

Creating and Scheduling a Campaign

Sequence diagram to depict messages for a user to create and schedule a campaign to run on the INTERSECT ecosystem is shown in Figure 5.5.

Running a Campaign

Sequence diagram to depict messages for a user to run a campaign to run on the INTERSECT ecosystem is shown in Figure 5.6.

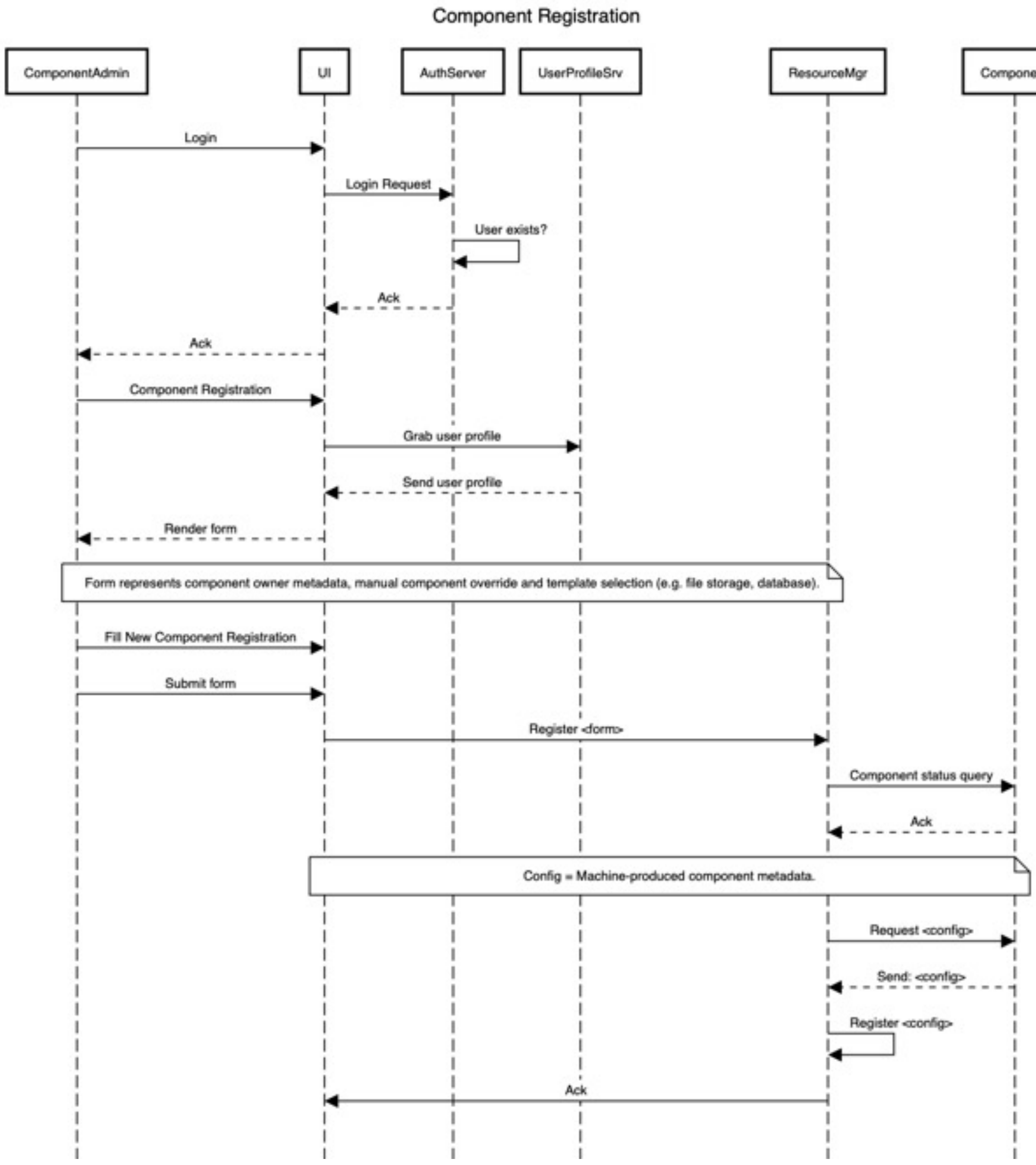


Figure 5.3. Sequence diagram for INTERSECT component registration.

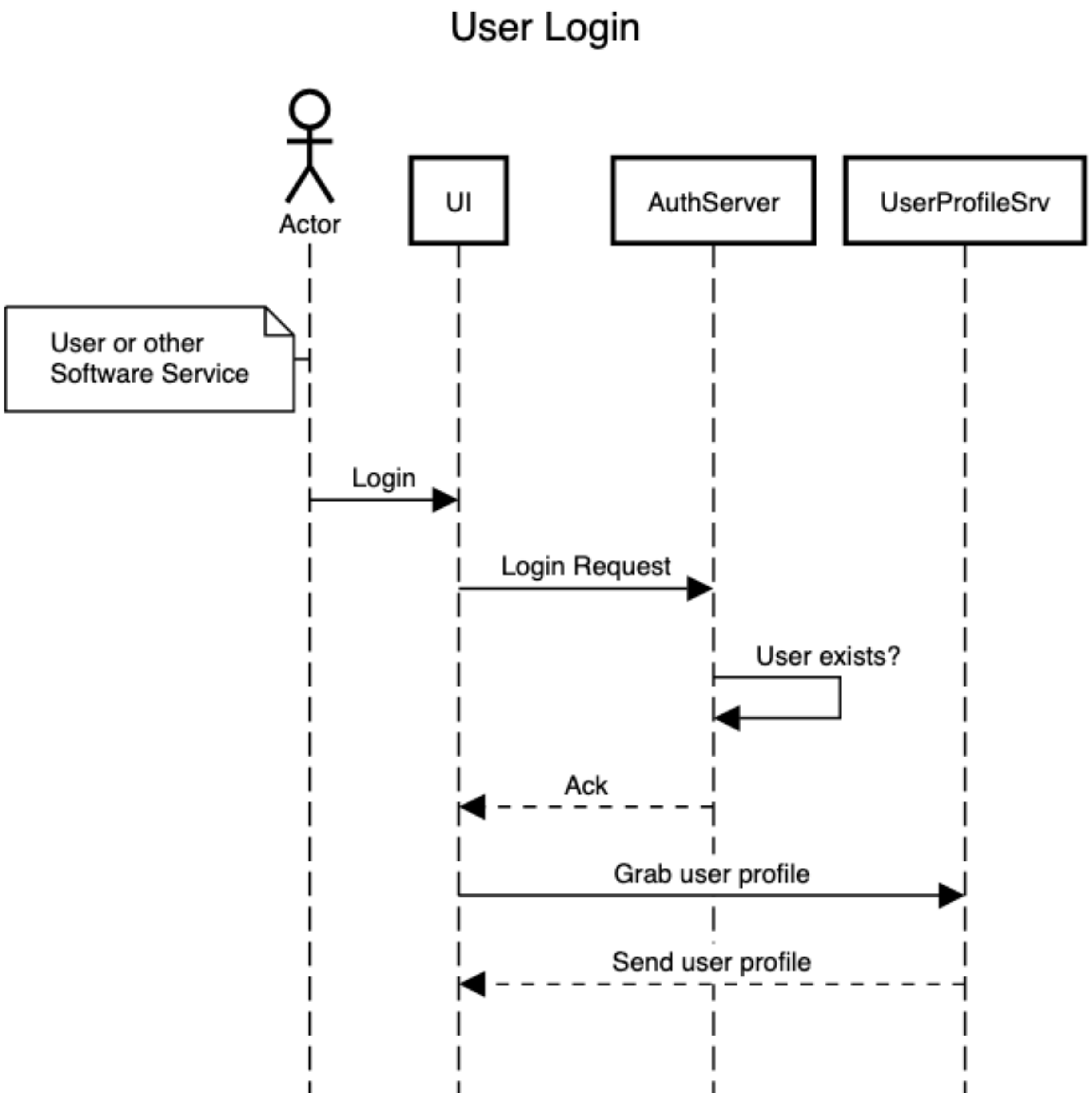


Figure 5.4. Sequence diagram for user login to INTERSECT.

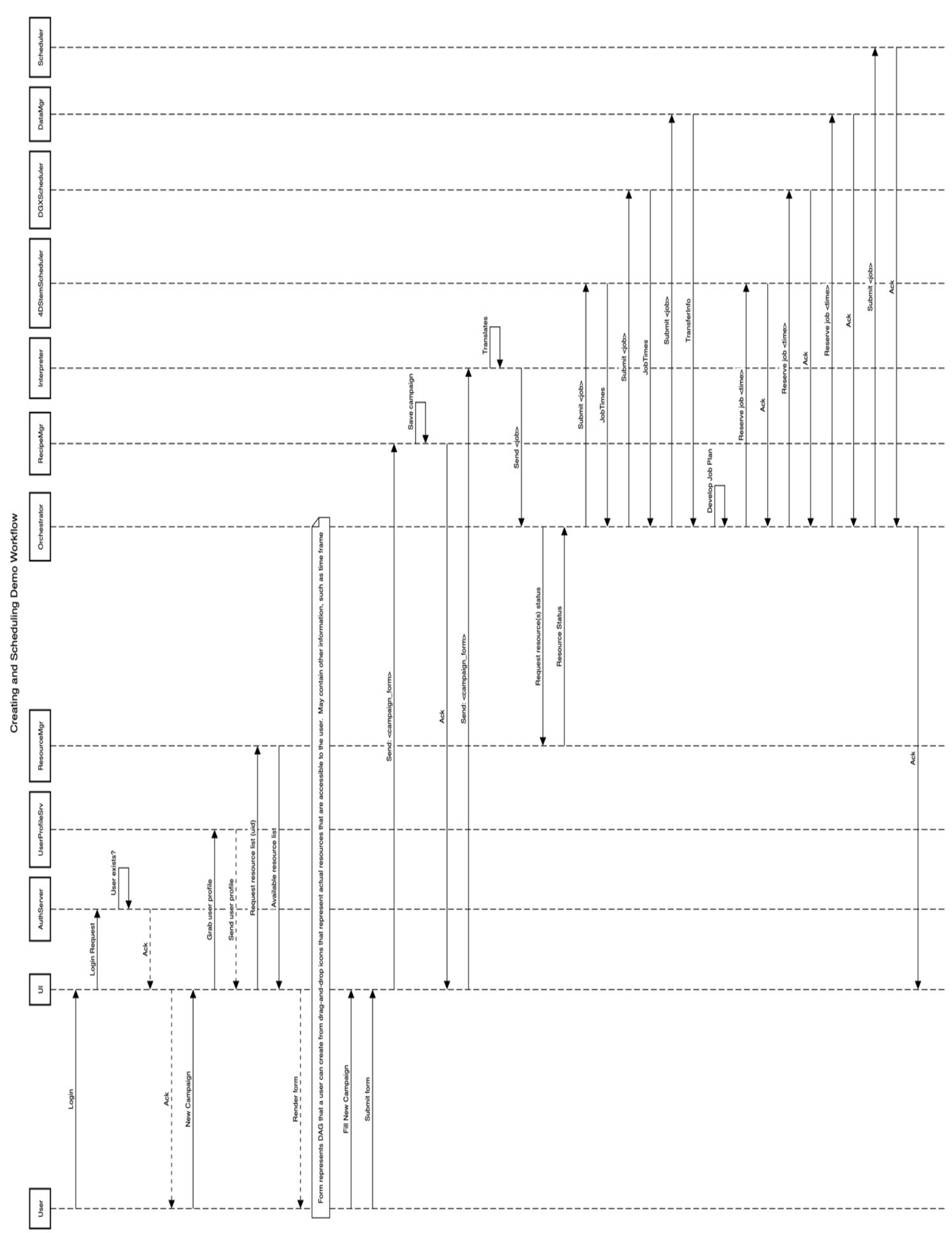


Figure 5.5. Sequence diagram for creating and scheduling a campaign on INTERSECT.

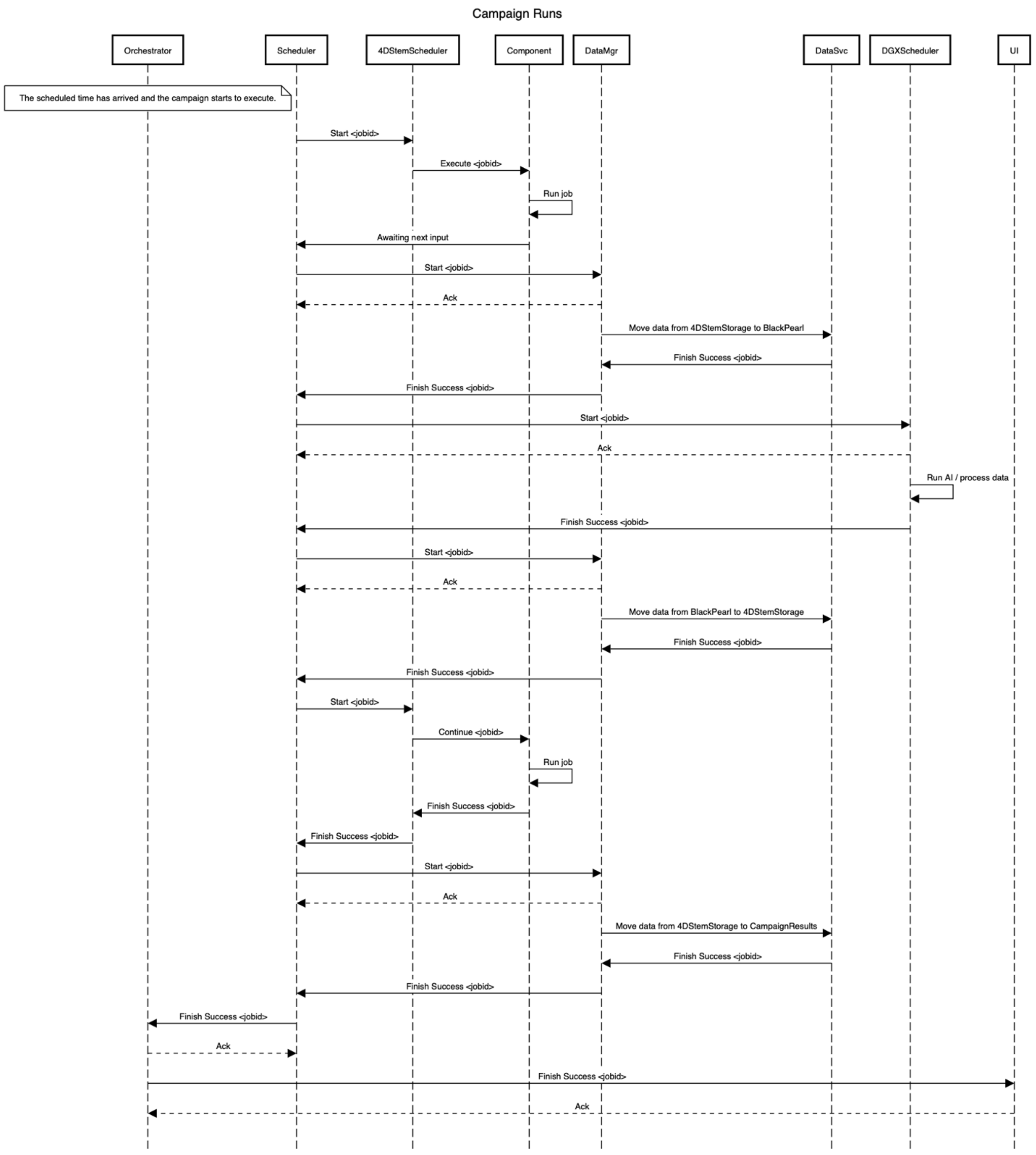


Figure 5.6. Running a campaign on INTERSECT.

6 PHYSICAL VIEW

This section of the architecture specification defines the Physical View.

6.1 INTRODUCTION

The physical view provides a mapping of the architecture onto the physical infrastructure. This view of the environment enables system designers to determine how to decompose and place the various system components onto the resources that make up the overall system. This view provides the architecture with an understanding of the attributes of the environment and allows the system to configure its services based on the constraints and capabilities of the underlying system components. The elements in this view consist of physical resources that provide services to the architecture as well as the network topology that connects them together. Types of physical resources include computational resources, data storage services, data sources, and network connections.

The physical view enables the enumeration of constraints placed on the overall system. These constraints can consist of capacity constraints (e.g., available storage capacity or computational elements), network constraints (e.g., available bandwidth between elements in the architecture), policy constraints (e.g., firewall rules or access control policies), and availability constraints (e.g., ability to allocate resources within necessary time frames). These constraints limit the configuration space of the architecture, and enumerate the necessary interfaces and processes required to configure the physical infrastructure to support the operations of the overall system.

What is in the physical view?

Some of the items associated with the physical view are:

- Descriptions, definitions of physical systems
- Descriptions of networks and connectivity of systems
- Descriptions of organizational boundaries

What is NOT in the physical view?

Some of the items associated with the physical view are:

- Specifications for instruments, resources, etc.
- Specifications for experiments
- Specifications for data
- Specifications for the logical view

Stakeholders and Concerns

This view should be read by resource managers/owners, system administrators, network engineers, and facility space managers. The phyiscal view is inherently site specific and provides a framework for describing the organization and technical challenges that must be addressed to deploy the architecture described in this document. The contents of this section will provide insight to stakholders to describe general challanges and concerns that must be addressed, as well as outline processes for fully developing the physical view for potential deployment sites.

Processes for Developing the Physical View

The nature of the INTERSECT architecture requires a highly collaborative approach in order to fully realize broad system integration across organizational boundaries. This collaboration must span multiple levels of the organization and will touch on a broad range of topics including specific procedural implementations, policy configurations and restrictions, organizational processes, and clear identification of organizational roles responsible for enabling cross organizational connections. The issues described by this view will cross multiple levels of the organizational hierarchy, and range from phyiscally configuring network connections between systems all the way to reconciling policies and procedures that govern usage of organizational resources.

Resource Taxonomy and Behavioral Model

Initial development of the physical view requires the development of a taxonomy of the resources that make up the environment of the organization being integrated with INTERSECT. This taxonomy should describe the classes of resources, what role they play within the environment, and the basic capabilities that each resource provides to the environment. This taxonomy allows a high level assessment of what components constituate a given system and provides a high level overview of which components are relevant to the INTERSECT system architecture. From this high level view, stakeholders can then decompose in an abstract way the various INTERSECT components and identify the relevant target system components they will need to interface with. This taxonomy should capture a high level description of the overall capability of the organization and environment to identify areas of concern that will likely require additional efforts to integrate with INTERSECT.

Endpoint Identification and Description

Following from the high level taxonomy, a full description of the current and future system environment needs to be collected. This description will include concrete description of actual physical resources that will need to be directly integrated with INTERSECT as API endpoints. The granularity of these descriptions should be at the interface boundary between INTERSECT and the target system. This description is separate from the taxonomy description in that it identifies specific system components and instances that need to interface with INTERSECT APIs. As an example, in the case of a computational cluster the taxonomy description would include the overall capabilities of the cluster itself (e.g. storage and compute capacity), while the endpoint description would include the cluster's head/login nodes and/or scheduler interface. The identification of the endpoints should be complete, i.e. it should fully capture all of the actual endpoints that are both currently active in the system as well as future endpoints that are planned for but not yet activated.

The purpose of this description is to provide a complete list of system components to be integrated into the system, and allows the identification of the responsible stakeholders that will need to be engaged with during the integration process.

Network Topology and Connectivity

After the INTERSECT endpoints have been identified, the next step is to collect information about the network topology of the target system. This topology should include both physical and functional connectivity as well as the performance capabilities of the underlying network infrastructure. Physical connecticity refers to the actual links (copper wires, optical fibers, or radio transmitters) that physically connect the INTERSECT endpoints. The topology does not need to include network interconnects that are internal to a system components, but should include any external connections to other system components. Using the compute cluster example again, the network topology would not need to include an internal Infiniband network with no external routes (this should be included as part of the taxonomy description) but should include the ethernet network that allows users to login to the head node. Functional connectivity refers to the ability of the network to support actual functional connections between system components. Functional connectity is different from physical connectivity due to the presence of routing rules, vlan configurations, or firewalls that interpose the physical connections between components. The purpose of separating these views of the network is to accurately identify the stakeholders to be engaged in order to alter or expand the current network topology. For instance, adding a routing rule will often require working with a different set of stakeholders than physically running new cables between systems. The description of the functional connectivity should include a full description of the components that enforce functional constraints, including fully identifying any firewalls, routers, or other middleboxes that might need to be reconfigured. These network resources may or may not be then added to the list of INTERSECT endpoints depending on whether they are reconfigurable or otherwise capable of interfacing with the INTERSECT architecture. Finally, the network topology should include the performance capabilities of the underlying network components. This is necessary in order to determine whther the available network resources are sufficient to meet the requirements of any potential INTERSECT workloads.

Security Policies and Access Control

Once the physical and network resources have been catalogued, the next step is to collect the security policies that govern how those resources may be accessed and how they can be used. These policies will take many forms, and do not necessarily need to be specified in exact detail. The important part is to identify the various points of constraint in the system and identify the relevant stakeholders that are responsible for managing those constraints. While not always possible, the most effective way of describing policies is to identify one or more constraints that are directly related, map those to the one or more mechanisms that enforce that policy, and then map the mechanims to the responsible stakeholders in the organization. This mapping allows the determination of any policy constraints that will impact the operation of INTERSECT and provide the set of stakeholders that need to be engaged with in order to manage any policy conflicts.

Access control is a major component of the security policies of a given system, but should also be considered separatly in the physical view. A description of the access control mechanisms should include the actual authentication procedures as well as a definition of the user set that is allowed to access the system components.

It is important to note that security policies are likely to not be fully global policies that uniformly apply to all resources in the system, but instead are often granular with specific constraints targetting specific components. In the INTERSECT physical view, it is important to differentiate the two and accurately identify the policies that apply to a particular INTERSECT endpoint.

Administrative Processes

Up to this point the physical view has focused on the actual physical infrastructure of the target environment, however it is important to also understand how that infrastructure is managed and administered by the organization. This requires understanding the processes that control who is able to make administrative decisions and how those decisions are made. This is a critical component because the INTERSECT architecture will most likely require administrative changes and allowances to enable it to effectively integrate with the environment. Depending on the organization in question these processes may be more or less well defined, but should nonetheless be identified as such in order to understand not only how to achieve integration with INTERSECT but also how to maintain that integration in the future. Similar to the previous aspects of the physical view, this description does not need to be exhaustive but should fully capture the processes that are required for managing the INTERSECT endpoints and connecting them to the wider INTERSECT architecture.

The purpose of this part of the description is to identify where the boundary at which INTERSECT stops interfacing with the physical environment and where it starts interfacing with the human level organizational systems. This separation indicates at which point INTERSECT integration needs to shift from software level interfaces to inter-organizational discussions and coordination in order to determine how to map the target organizations processes to the INTERSECT requirements. There are a number of avenues for this, including formalizing existing processes in such a way that they can be automated or at least automatically initiated, deploying federated policy models allowing the target organization to delegate administrative processes to other INTERSECT enabled organizations, or altering the actual technical integration approach to accomodate restrictions resulting from existing administrative requirements.

As an example of a basic administrative process that must be interfaced with is the user account and management processes employed by the organization. These processes govern how user accounts are created, how they are granted access to the various ogranizational resources, how access permisions are approved and authorized, and who maintains oversight of the accounts once created. For most INTERSECT enabled systems it can be expected that user account managment will be a fundamental challenge. Furthermore, while federation based approaches are being explored it is likely that target organizations will lack either the capability or desire to adopt such an approach.

Accounting Practices

Finally, the physical view needs to include a description of the accounting practices employed by the target organization. While this is similar to the description of the administrative processes, it is often the case that accounting procedures and models are based on more overarching and stringent processes that the organization itself has less control over. Accounting practices determine how usage of a given resource is monitored and accounted for and often, but not always, ties that usage to a cost model that governs how a given user is charged to use the system. Interfacing with these systems requires that INTERSECT include awareness of both the cost structure and accounting methods employed by the target environment. This is necessary to ensure that the target organization is able to correctly charge for the usage of its resources as well as providing transparency of the cost model to INTERSECT users and components.

7 STANDARDS VIEW

The INTERSECT SoS architecture incorporates various standards relating to specific instruments, messaging, and other external topics. The Standards View presented in this section provides a table of supported standards and other views or architecture elements that are impacted by each standard.

This section also defines a set of rules governing the arrangement, interaction, and interdependence of systems.

This section will be expanded in a future version of this document.

Bibliography

[1] Computational-Facilities. DOE national laboratories' computational facilities – Research workshop report. Technical Report ANL/MCS-TM-388, Argonne National Laboratory, Lemont, IL, USA, February 2020. URL https://publications.anl.gov/anlpubs/2020/02/158604.pdf.

[2] ISO/IEC/IEEE. Systems and software engineering – Architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pages 1 –46, 1 2011. doi: 10.1109/IEEESTD.2011.6129467.

[3] Kruchten, P. Architectural Blueprints – The "4+1" View Model of Software Architecture, 1995. URL http://www.cs.ubc.ca/~gregor/teaching/papers/4+1view-architecture.pdf.

[4] Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown. AI for science report, March 2020. URL https://www.anl.gov/ai-for-science-report.

[5] U.K. Ministry of Defense. The MOD architecture framework, September 14, 2011. URL http://www.modaf.org.uk/.

[6] U.S. Department of Defense. The DoDAF architecture framework version 2.02, August 2010. URL https://dodcio.defense.gov/Library/DoD-Architecture-Framework.

:

APPENDICES

A USER VIEW WIRE DIAGRAMS

This section provides a detailed listing of User View wire diagrams to outline usage from different viewpoints. These include the Owner, Operator and Administrator as described in Section 4.

A.1 USER PERSON TYPE AND ASSOCIATED VIEWS

This section will describe the different views associated with the User person type as described in Table 4.1. Any diagrams of user interfaces are only for informational purposes and are not intended to be actual GUIs.

User applies for an account on INTERSECT

Preconditions

User doesn't have an account on INTERSECT already but the User has all the information necessary to create an account, possibly including but not limited to a federated identity management account such as a Globus ID or a OneID account. INTERSECT web interface must be accessible and available to the general public. A user may also access INTERSECT via Command Line Interface (CLI) or via APIs.

Postconditions

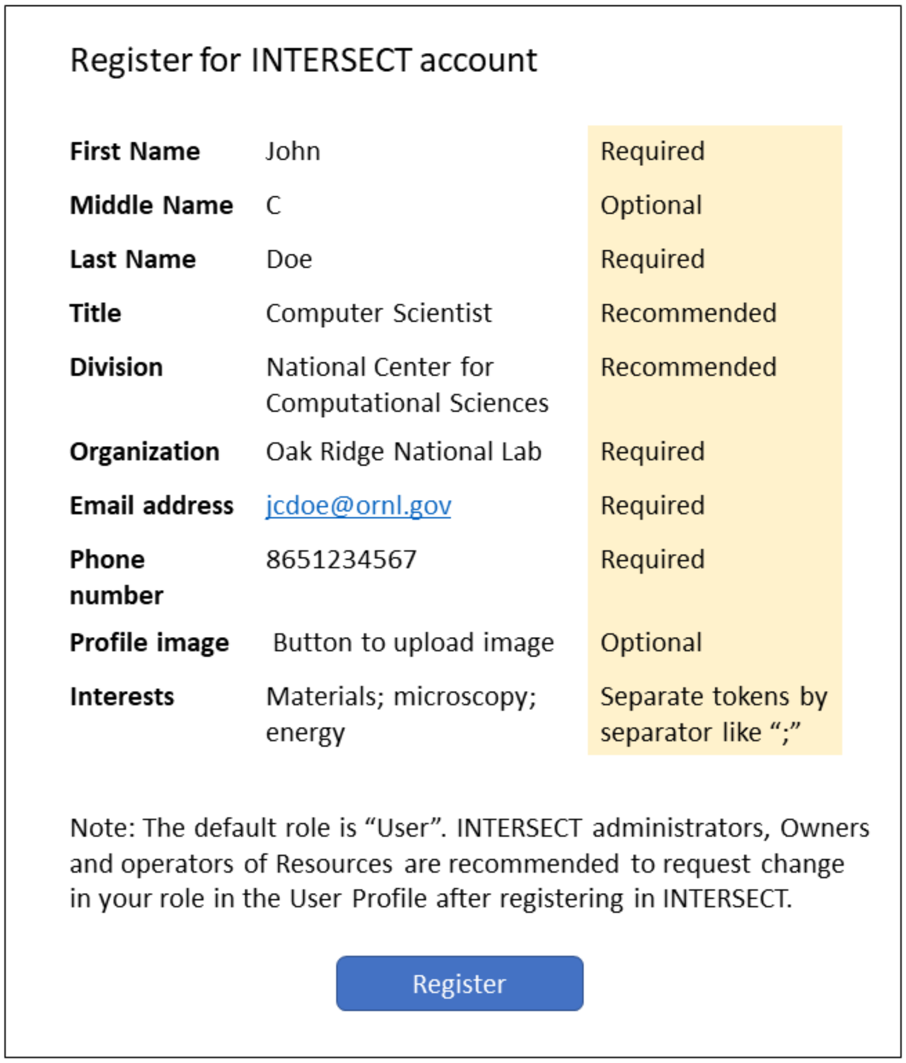
User should have an account in INTERSECT upon registration barring any email verification steps. User should be able to access INTERSECT capabilities and start composing a Campaign.

Methodologies

The User will be asked to provide the following information to get an account on INTERSECT:

- Full Name – required
- username – required
- Organization – required
- Facility – required
- Division – required
- Title in organization – required
- Email address – required
- Phone number – optional
- Interests – recommended
- Profile photograph – optional

An example INTERSECT account application is depicted in Figure A.1.



Register for INTERSECT account

First Name	John	Required
Middle Name	C	Optional
Last Name	Doe	Required
Title	Computer Scientist	Recommended
Division	National Center for Computational Sciences	Recommended
Organization	Oak Ridge National Lab	Required
Email address	jcdoe@ornl.gov	Required
Phone number	8651234567	Required
Profile image	Button to upload image	Optional
Interests	Materials; microscopy; energy	Separate tokens by separator like ";"

Note: The default role is "User". INTERSECT administrators, Owners and operators of Resources are recommended to request change in your role in the User Profile after registering in INTERSECT.

Figure A.1. A representation of the information that a new or potential user of INTERSECT would be required to register for an INTERSECT account..

User logs into INTERSECT Dashboard

Preconditions

User has already registered with an account in INTERSECT. User has an account with the chosen federated identity management provider.

Postconditions

Upon successfully logging in, the User can now access the INTERSECT dashboard.

Methodologies

User logs into INTERSECT via the login screen

- Auth method (ORCID, Globus, OneID) >> Institution >> username >> password

An example user INTERSECT dashboard is depicted in Figure A.2.

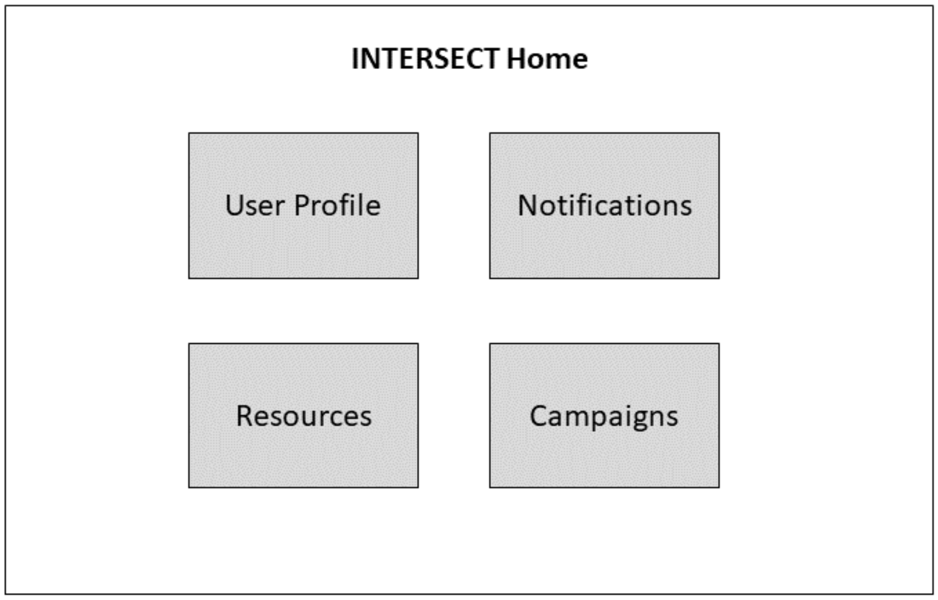


Figure A.2. A representation of the home-screen that would be displayed once a User logs into the INTERSECT web interface..

User Explores the INTERSECT Dashboard

Preconditions

User has successfully logged into INTERSECT

- User is shown tiles or links to the following the main panels:
  - User Profile
  - Resource Catalog
  - Campaign Listing
  - Notification Panel

Postconditions

User has access to view Resources, their past, present, and future Campaigns, notifications, etc.

User Profile

Preconditions

User is registered and logged in successfully.

Postconditions

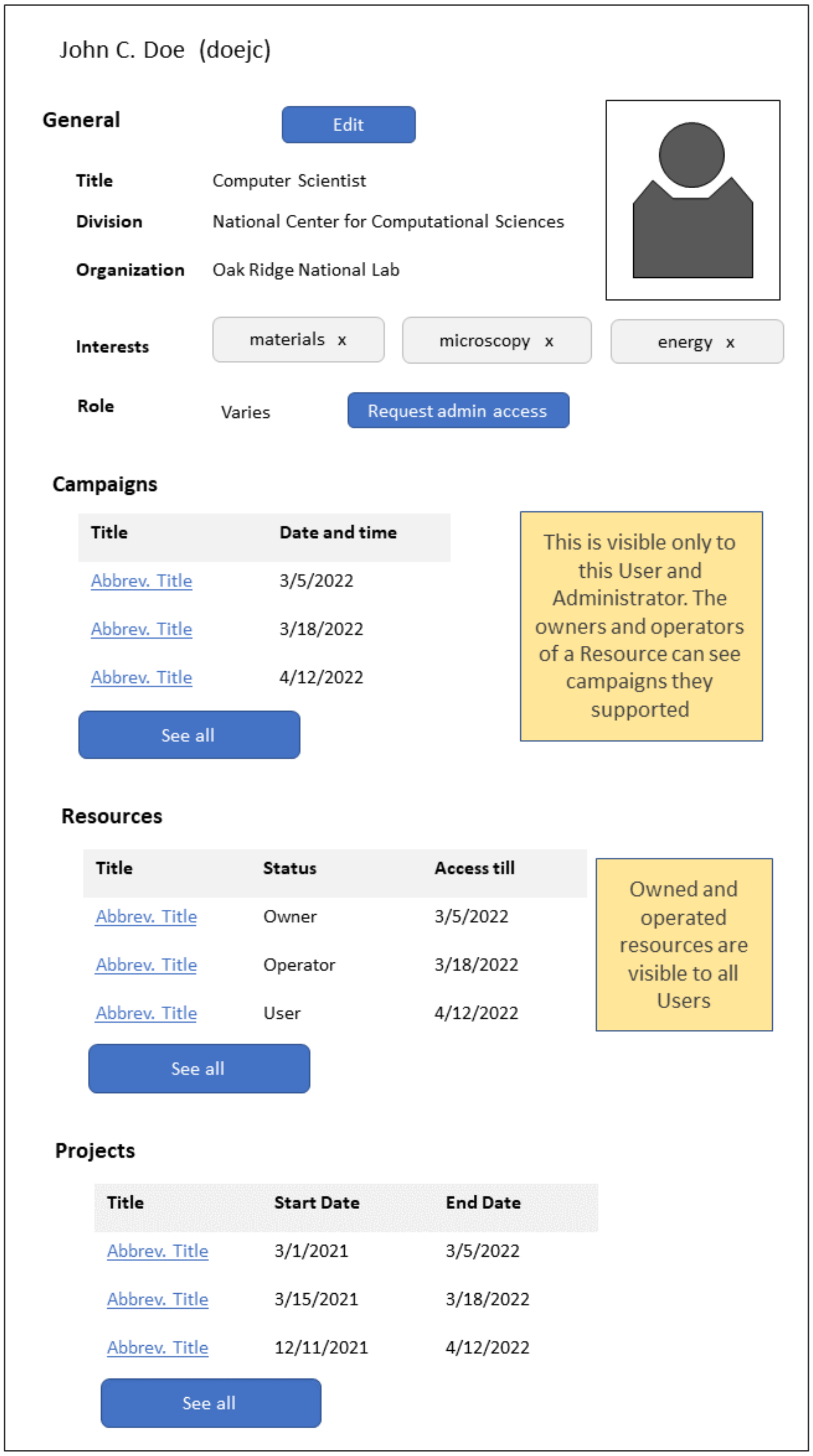
User can view all details about their profile and be capable of making changes as necessary.

Methodologies

The User Profile is something that others could also probably see to get an understanding about this User. It would have the following details:

- Username
- Full name
- Thumbnail image of person
- Organization
- Facility within Organization
- Title in Organization
- Role in INTERSECT
- Interests
- Campaign listings (perhaps only the past ones are visible to other Users – see below)
- Resources managed / owned by this User
  - this Owner / Operator gets to see a listing of everything they own / maintain
  - If this User were a Maintainer / Operator of multiple Resources
    - only those Resource(s) owned by the Owner, who is viewing the page, will be displayed.
    - Other regular Users will not be able to see these?
  - an Owner of the Resource(s) can see those Resource(s) listed only relevant to Owner / Maintainer)
- A button to edit certain basic details about their profile

An example user profile is depicted in Figure A.3.


John C. Doe (doejc)

General

Title	Computer Scientist
Division	National Center for Computational Sciences
Organization	Oak Ridge National Lab
Interests	materials x, microscopy x, energy x
Role	Varies

Campaigns

Title	Date and time
Abbrev. Title	3/5/2022
Abbrev. Title	3/18/2022
Abbrev. Title	4/12/2022

This is visible only to this User and Administrator. The owners and operators of a Resource can see campaigns they supported

Resources

Title	Status	Access till
Abbrev. Title	Owner	3/5/2022
Abbrev. Title	Operator	3/18/2022
Abbrev. Title	User	4/12/2022

Owned and operated resources are visible to all Users

Projects

Title	Start Date	End Date
Abbrev. Title	3/1/2021	3/5/2022
Abbrev. Title	3/15/2021	3/18/2022
Abbrev. Title	12/11/2021	4/12/2022

Figure A.3. A representation of a registered User's profile in INTERSECT with options to edit their profile.

User Profile

Preconditions

User is registered and logged into INTERSECT

Postconditions

Changes in user details now visible to this User and to any other User in INTERSECT.

Methodologies

- Upon clicking the edit button in the User Profile, boxes containing the following fields become editable:
  - Full Name
  - Title / Role in Organization
  - Division
  - Organization
- The user clicks on the “Save” button to commit any changes they made.

User Views Project

Preconditions

User is registered and logged into INTERSECT. User is part of the project being viewed or User is an INTERSECT administrator.

Postconditions

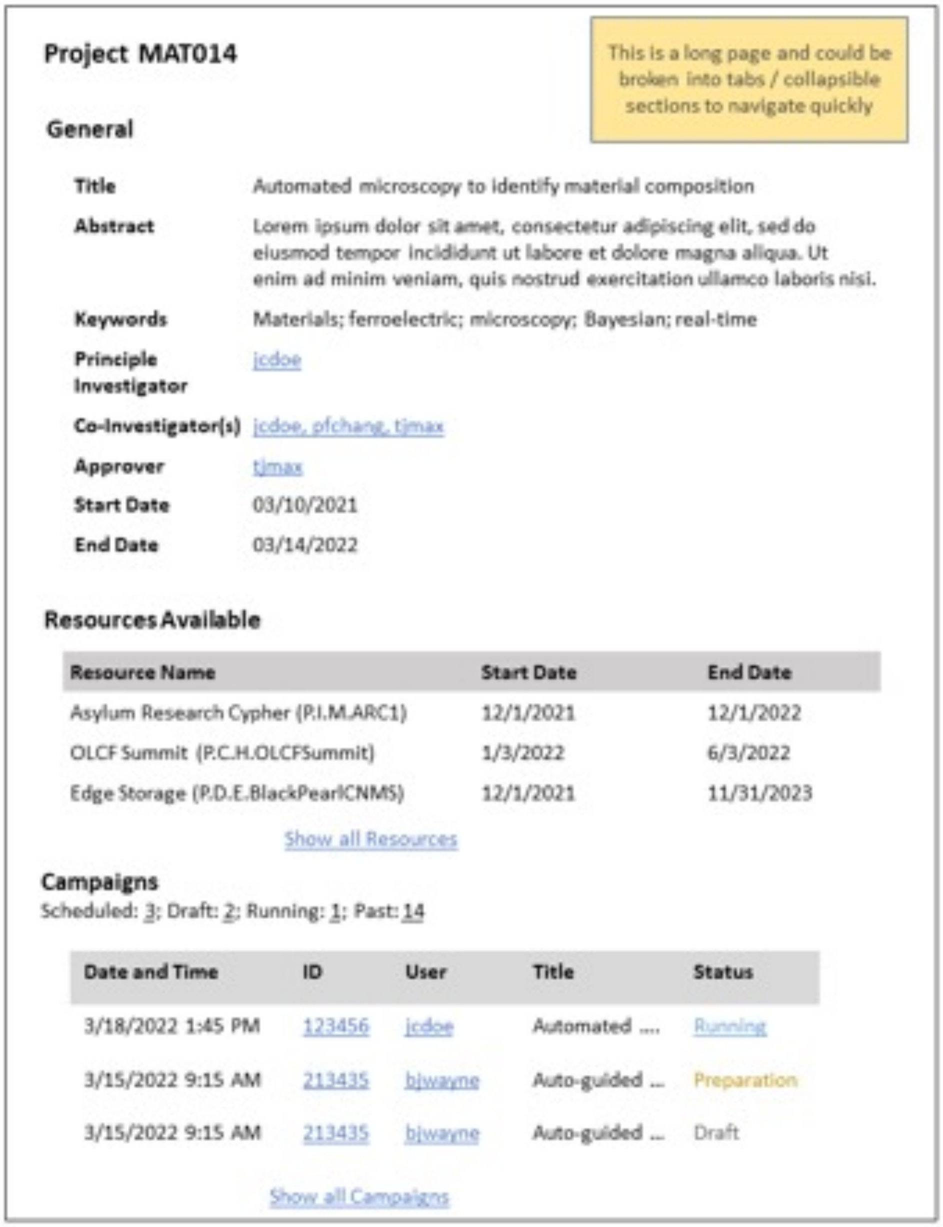
User has knowledgeable about the project

Methodologies

This view should show information about the Project with:

- Basic information
  - Title
  - Abstract
  - Keywords
  - Principle investigator, co-investigator, others
  - Approvers for campaigns
  - Start and end date for Project
- Resources available to Project
- Campaigns (past, present, and future) that are part of this project.

An example interface is depicted in Figure A.4.


Project MAT014

This is a long page and could be broken into tabs / collapsible sections to navigate quickly

General

Title	Automated microscopy to identify material composition
Abstract	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi.
Keywords	Materials; ferroelectric; microscopy; Bayesian; real-time
Principle Investigator	jcdoe
Co-Investigator(s)	jcdoe, pfchang, tjmax
Approver	tjmax
Start Date	03/10/2021
End Date	03/14/2022

Resources Available

Resource Name	Start Date	End Date
Asylum Research Cypher (P.I.M.ARC1)	12/1/2021	12/1/2022
OLCF Summit (P.C.H.OLCFSummit)	1/3/2022	6/3/2022
Edge Storage (P.D.E.BlackPearlCNMS)	12/1/2021	11/31/2023

Show all Resources

Campaigns

Scheduled: 3; Draft: 2; Running: 1; Past: 14

Date and Time	ID	User	Title	Status
3/18/2022 1:45 PM	123456	jcdoe	Automated	Running
3/15/2022 9:15 AM	213435	bjwayne	Auto-guided ...	Preparation
3/15/2022 9:15 AM	213435	bjwayne	Auto-guided ...	Draft

Show all Campaigns

Figure A.4. A representation of a User's project view..

User Views List of Campaigns

Preconditions

User is registered and logged into INTERSECT

Postconditions

User has knowledge about the Campaigns they are involved in, in addition to being able to create a new Campaign.

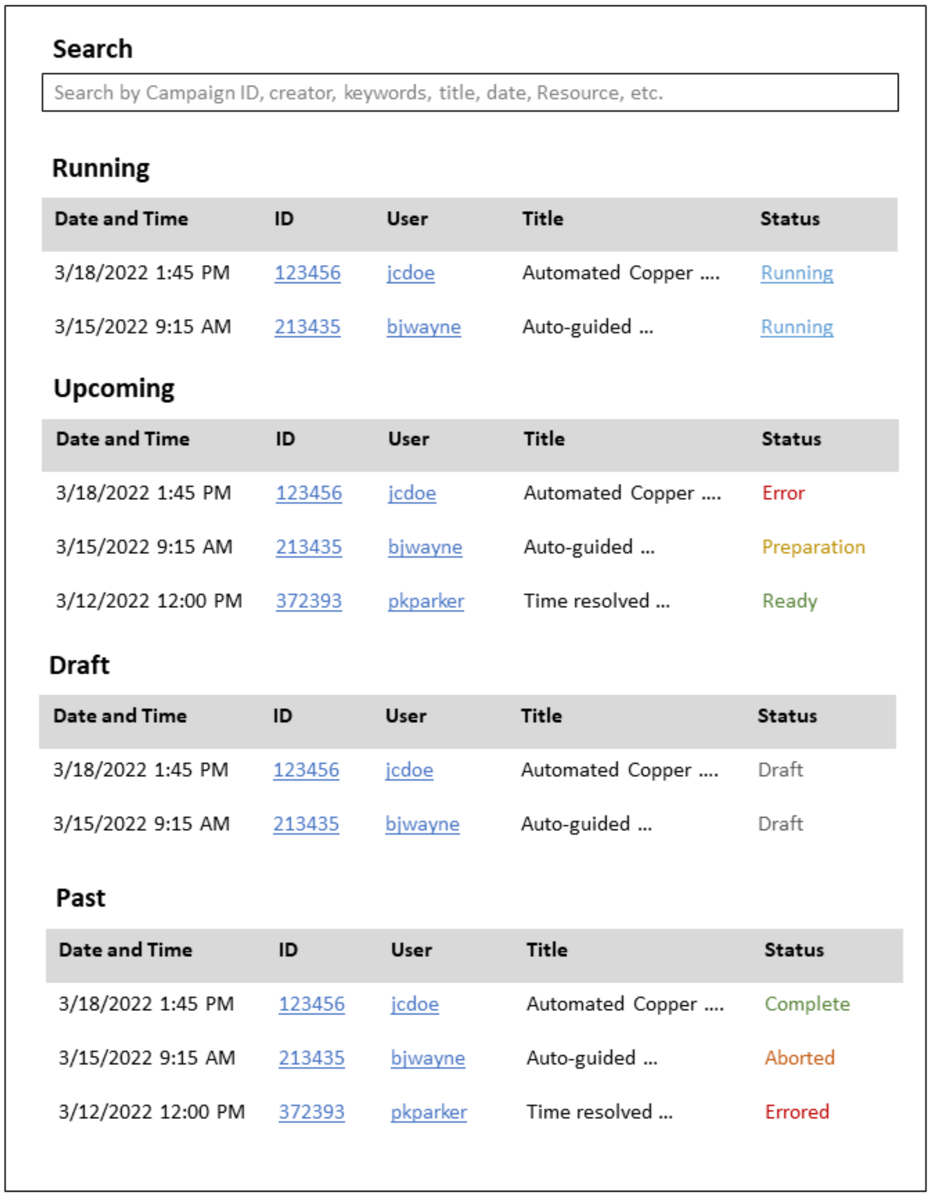
Methodologies

This view should be a listing view with the following information per Campaign:

- Date and time of Campaign
- Campaign ID
- Link to view more details about the campaign
- User who is the primary point of contact for this Campaign
- Abbreviated / short title
- Status - this field would be represented differently based on the state of the Campaign:
  - Ongoing Campaigns - link to jump directly into the live monitoring dashboard for the Campaign
  - Scheduled / upcoming Campaigns - aggregated or lowest common denominator status of all major resources, and processes (E.g. approval from PI), marked by symbols and colors. This would be a single value denoting the weakest link in the chain.
  - Draft - Status: marked as Draft. No options / variations

This view should also provide the User the ability to search for Campaigns by the original composer (User), title, keywords, date, etc.

An example interface is depicted in Figure A.5.



Search

Search by Campaign ID, creator, keywords, title, date, Resource, etc.

Running

Date and Time	ID	User	Title	Status
3/18/2022 1:45 PM	123456	jcdoe	Automated Copper	Running
3/15/2022 9:15 AM	213435	bjwayne	Auto-guided ...	Running

Upcoming

Date and Time	ID	User	Title	Status
3/18/2022 1:45 PM	123456	jcdoe	Automated Copper	Error
3/15/2022 9:15 AM	213435	bjwayne	Auto-guided ...	Preparation
3/12/2022 12:00 PM	372393	pkparker	Time resolved ...	Ready

Draft

Date and Time	ID	User	Title	Status
3/18/2022 1:45 PM	123456	jcdoe	Automated Copper	Draft
3/15/2022 9:15 AM	213435	bjwayne	Auto-guided ...	Draft

Past

Date and Time	ID	User	Title	Status
3/18/2022 1:45 PM	123456	jcdoe	Automated Copper	Complete
3/15/2022 9:15 AM	213435	bjwayne	Auto-guided ...	Aborted
3/12/2022 12:00 PM	372393	pkparker	Time resolved ...	Errored

Figure A.5. A high-level view representing past, present, and future scientific Campaigns that this user has been part of.

User Creates Campaign

Preconditions

User can select a Campaign to view. This could be a template, a publicly visible completed Campaign, or one the User has taken part in / is taking part in.

Postconditions

User can understand details of the given Campaign

Methodologies

Text here

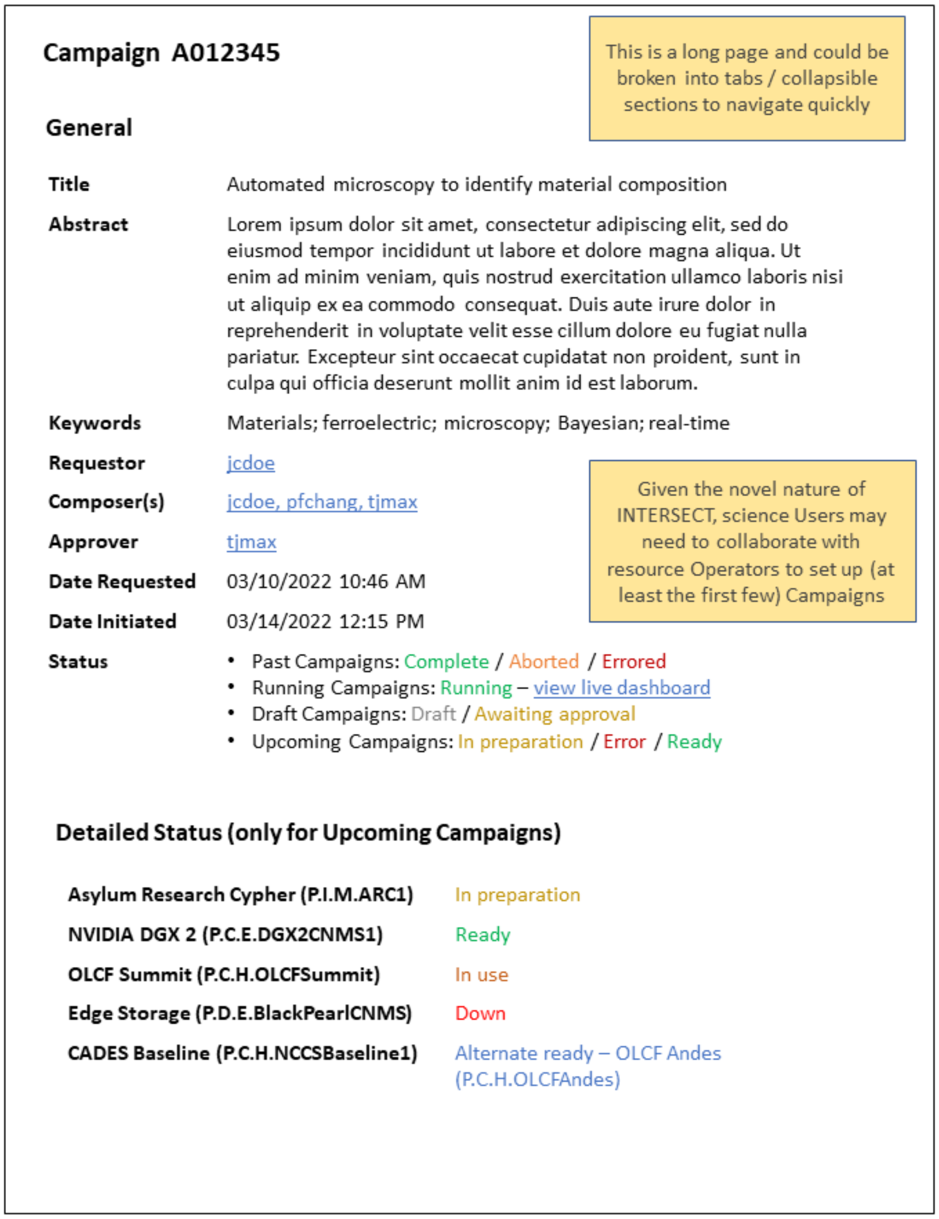
- Top level
  - User(s) / composer(s) of Campaign
    - Listing of Users - These Users have edit access to this Campaign and may collaboratively compose this Campaign
    - Approver - single User
      - This person who approved the Requestor's (e.g. PhD student / post-doc) submission of Campaign to INTERSECT
    - Requestor - User who requested this Campaign. Also, generally the primary person who monitors the Campaign
  - Date, time
    - Requested
    - Actual - only relevant for Past and Running Campaigns
  - Title
  - Domain classifier(s): E.g. Materials, Biology, ...
  - Abstract
  - Status
    - Past Campaigns - "Complete" or "Aborted" or "Errored"
    - Future Campaigns - a status symbol for each of the resources along with what is happening. E.g. - "Instrument being configured"
      - Ready - green check mark
      - Being prepared - yellow spinning progress icon
      - In use - gray exclamation mark
      - Down - red exclamation mark or cross sign
      - Alternate ready - blue check mark
    - Running Campaigns - Link / coordinates to attach something else - think OpenShift console - network coordinates
    - Draft - "Draft" or "Submitted" or "Under Review"
- Configuration
  - Workflow configuration (directed graph view?)
    - Parameter values
  - Requested resources
    - Primary
    - Secondary / backup for critical resources

- Resources used (physical ones could be different from the ones that were requested due to unavailability of resources)
    - Provider / Operators who participated where relevant
    - Properties and configuration
    - Calibration setting files if relevant
- Run-time dashboard widget setup
    - Widget placement and configuration?
- Periodic push notifications:
    - Mode: SMS, email, other?
    - Frequency: e.g. every 1 hour.
    - What to send as payload:
        - Text based information - e.g. -
            1. progress towards goal
            2. unexpected event - warnings
            3. catastrophic failure
            4. Campaign complete
            5. One or more high-level metrics that indicates health / progress of Campaign - e.g. battery life of remote sensor, volume of reactants in tanks / vials
        - output of a specific visualization
        - “screenshot” of primary real-time dashboard
        - Exports of simplified dashboard(s)
- Download / reuse for another Campaign

- Results
    - Running Campaigns only - Coordinates to attach something else - think OpenShift console - network coordinates
    - Completed and running Campaigns only - Persistent identifier for dataset (multiple files, full stream, etc.) in a data management system.
    - Running and completed Campaigns - Link to live dashboard for monitoring data
    - Completed Campaigns -
        - User would be able to scrub through time to observe evolution of data -perhaps replace or supplement the progress bar with a scrubber.
        - User would be able to export content from the configured plots to static image files
        - User would also be able to replace variables being plotted to observe the evolution of other factors during the Campaign
- Errors - listing of
    - Examples:
        - Network failure
        - Data stream failure
        - Globus unresponsive or endpoint not activated or expired
        - Error encountered at Instrument
        - Divergence error - if trying to match measurement and simulation
        - Safety error - chemical combination that may result in an unsafe work environment
        - Value out of range either from a sensor or an AI agent
        - Computational job failure
        - Resource failure with no contingency / redundancy available

- Warnings - listing of
  - For each Warning, the User can:
    - Dismiss it
    - Take action
    - Request Maintainer / Operator / Administrator to take action
  - Examples of warnings (mostly real-time):
    - Low battery in sensor
    - Low raw ingredient - e.g. chemical in vial
    - Unable to read telemetry from a secondary / non-critical sensor
    - Network bandwidth nearing set threshold
    - User steered Campaign in a manner that could potentially be dangerous
    - User steered Campaign in a manner that would degrade performance metric
- Logs
  - When Campaign was first composed
  - When Campaign was requested
  - Who requested the Campaign
  - Who (e.g. PI of research effort) approved the Campaign submission to INTERSECT and when
  - When each resource was marked at "being set up"
  - When each resource was marked as "ready"
  - When resources were swapped for alternate / backup
  - When Operator / Maintainer set up a resource + notes if any
  - When user logged back in to initiate Campaign
  - When the user logged back in to monitor the Campaign via the dashboard
  - Any on-the-fly change in parameters during the Campaign - what and when
  - When backup / redundant resources were used to compensate for insufficient performance by the primary resource
  - Warnings and errors in chronological order + what the user did about this
  - Maybe telemetry and other simple status information from Campaign to indicate its health
  - When Campaign ended - due to one of many factors
  - Where data was saved
- Button to use this as a template for new Campaign - not visible to Campaigns in draft mode / only visible to Campaigns that have been submitted and approved

An example of creating campaigns is depicted in Figure A.6 and Figure A.7.



Campaign A012345

This is a long page and could be broken into tabs / collapsible sections to navigate quickly

General

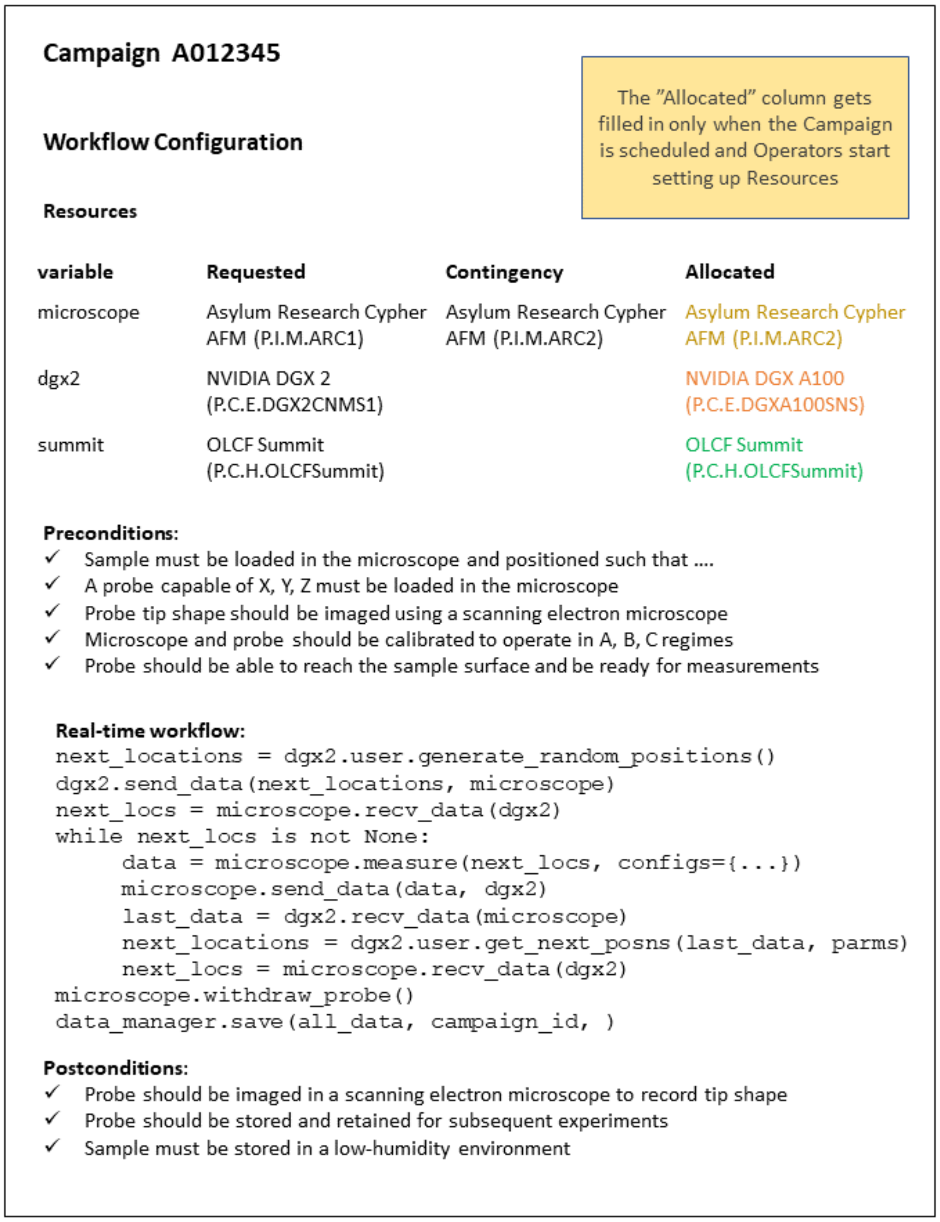
Title	Automated microscopy to identify material composition
Abstract	Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
Keywords	Materials; ferroelectric; microscopy; Bayesian; real-time
Requestor	jcdoe
Composer(s)	jcdoe, pfchang, tjmax
Approver	tjmax
Date Requested	03/10/2022 10:46 AM
Date Initiated	03/14/2022 12:15 PM
Status	• Past Campaigns: Complete / Aborted / Errored • Running Campaigns: Running – view live dashboard • Draft Campaigns: Draft / Awaiting approval • Upcoming Campaigns: In preparation / Error / Ready

Given the novel nature of INTERSECT, science Users may need to collaborate with resource Operators to set up (at least the first few) Campaigns

Detailed Status (only for Upcoming Campaigns)

Asylum Research Cypher (P.I.M.ARC1)	In preparation
NVIDIA DGX 2 (P.C.E.DGX2CNMS1)	Ready
OLCF Summit (P.C.H.OLCFSummit)	In use
Edge Storage (P.D.E.BlackPearlCNMS)	Down
CADES Baseline (P.C.H.NCCSBaseline1)	Alternate ready – OLCF Andes (P.C.H.OLCFAndes)

Figure A.6. Detailed view of a single scientific Campaign as viewed by a participant (composer, Owner or Operator of a requested Resource, Administrator) of the Campaign. This view focuses on the general information and detailed status of the primary Resources requested in this Campaign (once the Campaign is submitted)



Campaign A012345

Workflow Configuration

The "Allocated" column gets filled in only when the Campaign is scheduled and Operators start setting up Resources

Resources

variable	Requested	Contingency	Allocated
microscope	Asylum Research Cypher AFM (P.I.M.ARC1)	Asylum Research Cypher AFM (P.I.M.ARC2)	Asylum Research Cypher AFM (P.I.M.ARC2)
dgx2	NVIDIA DGX 2 (P.C.E.DGX2CNMS1)		NVIDIA DGX A100 (P.C.E.DGXA100SNS)
summit	OLCF Summit (P.C.H.OLCFSummit)		OLCF Summit (P.C.H.OLCFSummit)

Preconditions:

- ✓ Sample must be loaded in the microscope and positioned such that ....
- ✓ A probe capable of X, Y, Z must be loaded in the microscope
- ✓ Probe tip shape should be imaged using a scanning electron microscope
- ✓ Microscope and probe should be calibrated to operate in A, B, C regimes
- ✓ Probe should be able to reach the sample surface and be ready for measurements

Real-time workflow:

```
next_locations = dgx2.user.generate_random_positions()
dgx2.send_data(next_locations, microscope)
next_locs = microscope.recv_data(dgx2)
while next_locs is not None:
     data = microscope.measure(next_locs, configs={...})
     microscope.send_data(data, dgx2)
     last_data = dgx2.recv_data(microscope)
     next_locations = dgx2.user.get_next_posns(last_data, parms)
     next_locs = microscope.recv_data(dgx2)
microscope.withdraw_probe()
data_manager.save(all_data, campaign_id, )
```

Postconditions:

- ✓ Probe should be imaged in a scanning electron microscope to record tip shape
- ✓ Probe should be stored and retained for subsequent experiments
- ✓ Sample must be stored in a low-humidity environment

Figure A.7. Detailed view of a single scientific Campaign as viewed by a participant (composer, Owner or Operator of a requested Resource, Administrator) of the Campaign. This view focuses on the workflow configuration for the Campaign.

Catalog of Campaign Templates

Preconditions

User is registered and logged into INTERSECT.

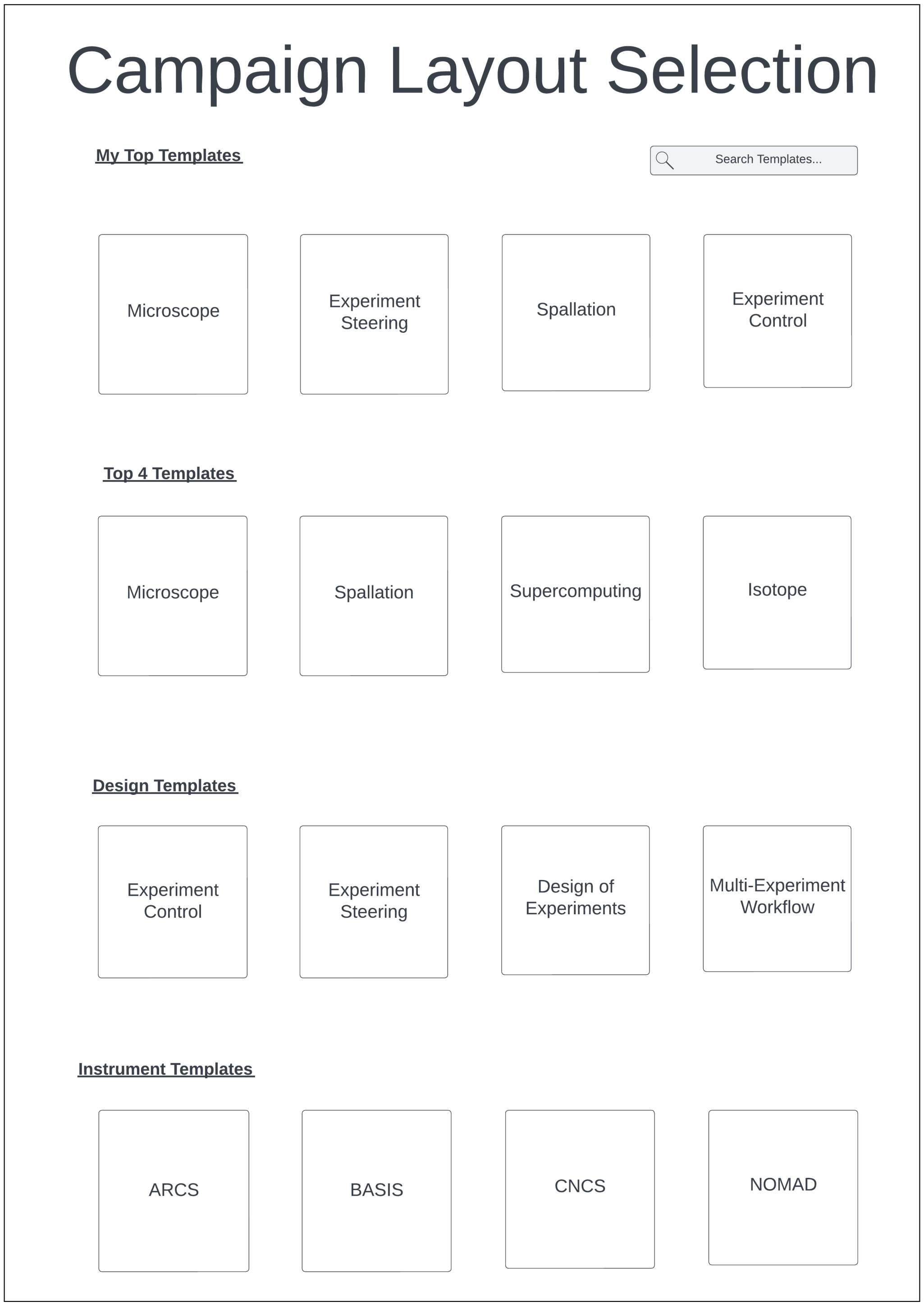
Postconditions

User has knowledge of starter templates to use as a basis for composing Campaigns.

Methodologies

- Listing view: Thumbnails (if available) with a title
- Detailed view of a selected campaign template:
  - Title
  - Intent
  - (Scientific) Background
  - Description (of how the intent is accomplished with this template)
  - Workflow:
    - Either a text based description with a static representation of the workflow (graph)
    - Or a non-editable version of the 100% graphical workflow composer view.
  - Recommended resources: 1-2 resources per component where there are options.
  - Listing of past campaigns that used this template
    - Date
    - User
    - Title (abbreviated) with link
  - "Use" or "Use Template" button that composes a campaign using this template

An example interface is depicted in Figure A.8 and Figure A.9


Campaign Layout Selection

My Top Templates

Search Templates...

Microscope

Experiment Steering

Spallation

Experiment Control

Top 4 Templates

Microscope

Spallation

Supercomputing

Isotope

Design Templates

Experiment Control

Experiment Steering

Design of Experiments

Multi-Experiment Workflow

Instrument Templates

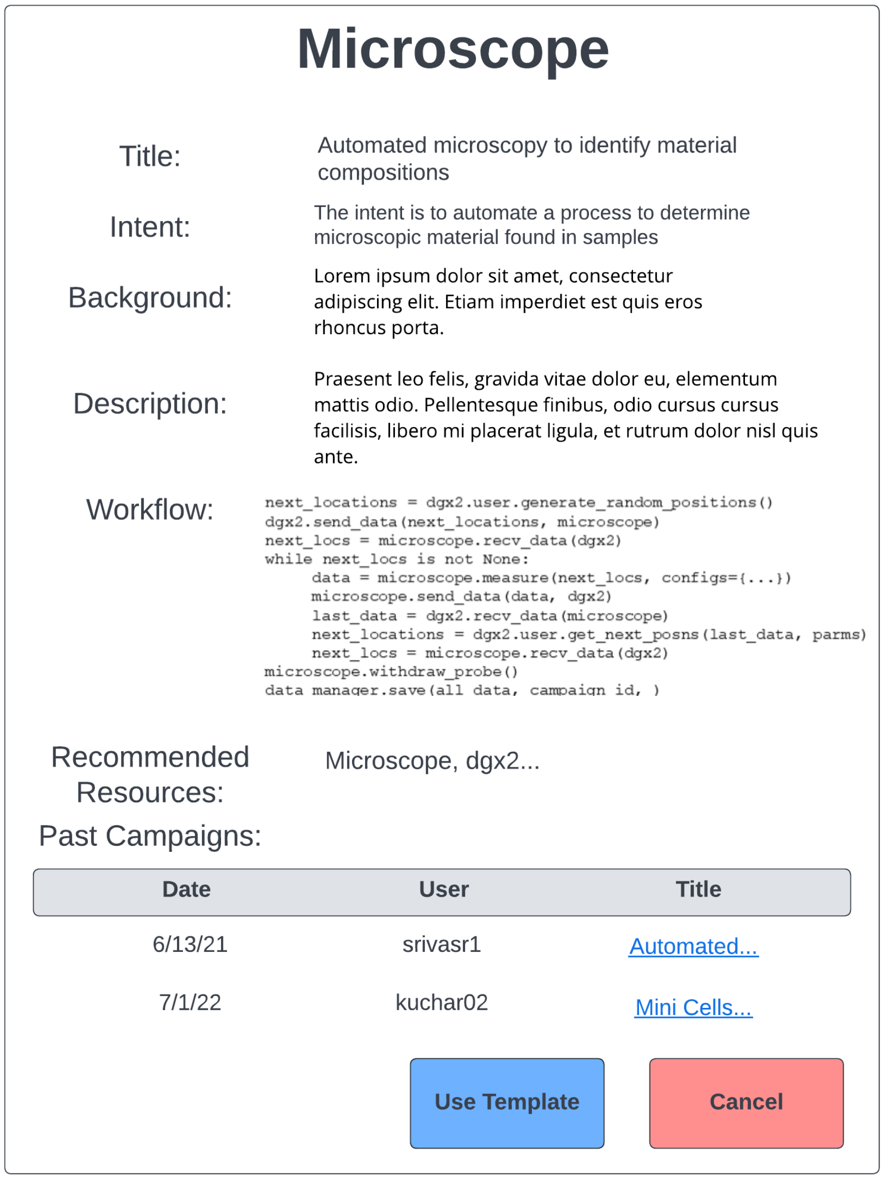
ARCS

BASIS

CNCS

NOMAD

Figure A.8. Buttons are shown for different campaign types.



Microscope

Title: Automated microscopy to identify material compositions

Intent: The intent is to automate a process to determine microscopic material found in samples

Background: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam imperdiet est quis eros rhoncus porta.

Description: Praesent leo felis, gravida vitae dolor eu, elementum mattis odio. Pellentesque finibus, odio cursus cursus facilisis, libero mi placerat ligula, et rutrum dolor nisl quis ante.

Workflow:

```
next_locations = dgx2.user.generate_random_positions()
dgx2.send_data(next_locations, microscope)
next_locs = microscope.recv_data(dgx2)
while next_locs is not None:
     data = microscope.measure(next_locs, configs={...})
     microscope.send_data(data, dgx2)
     last_data = dgx2.recv_data(microscope)
     next_locations = dgx2.user.get_next_posns(last_data, parms)
     next_locs = microscope.recv_data(dgx2)
microscope.withdraw_probe()
data_manager.save(all_data, campaign_id, )
```

Recommended Resources: Microscope, dgx2...

Past Campaigns:

Date	User	Title
6/13/21	srivasr1	Automated...
7/1/22	kuchar02	Mini Cells...

Figure A.9. When a campaign is selected, a basic template is shown.

Service Catalog

Preconditions

User is registered and logged into INTERSECT

Postconditions

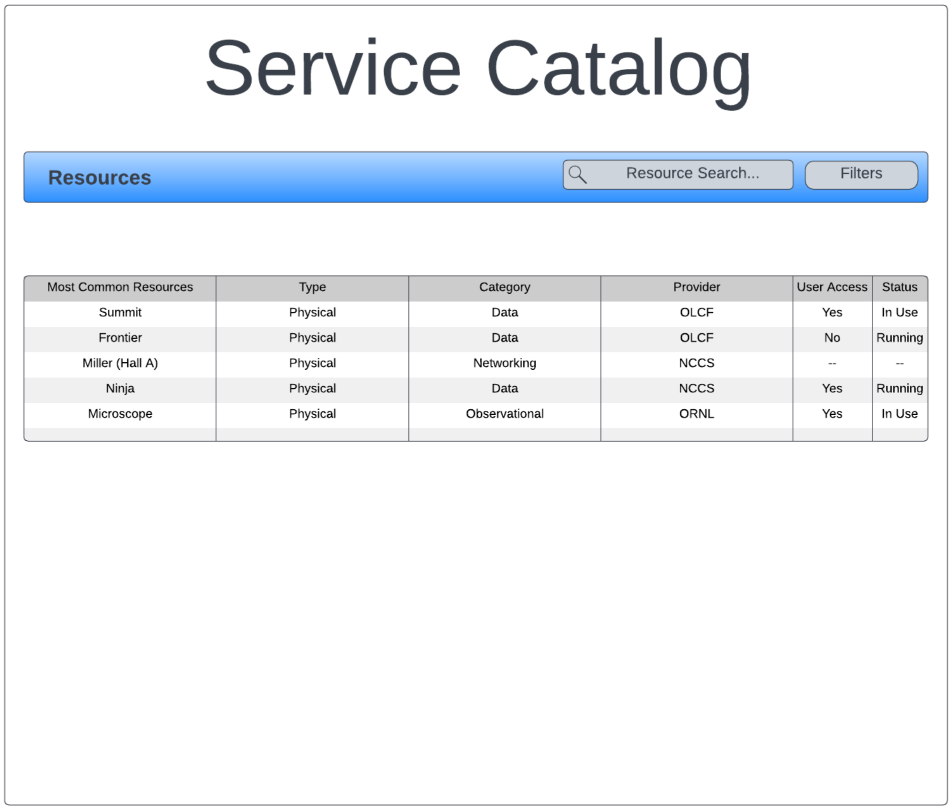
User can view all available resources in INTERSECT and choose to use discovered Resources in subsequent Campaigns.

Methodologies

This view should show a grid or listing of Resources available in INTERSECT along with tools to search for or filter Resources to efficiently find Resource(s) of interest. Clicking on any resource in the list would provide a detailed view of a given resource

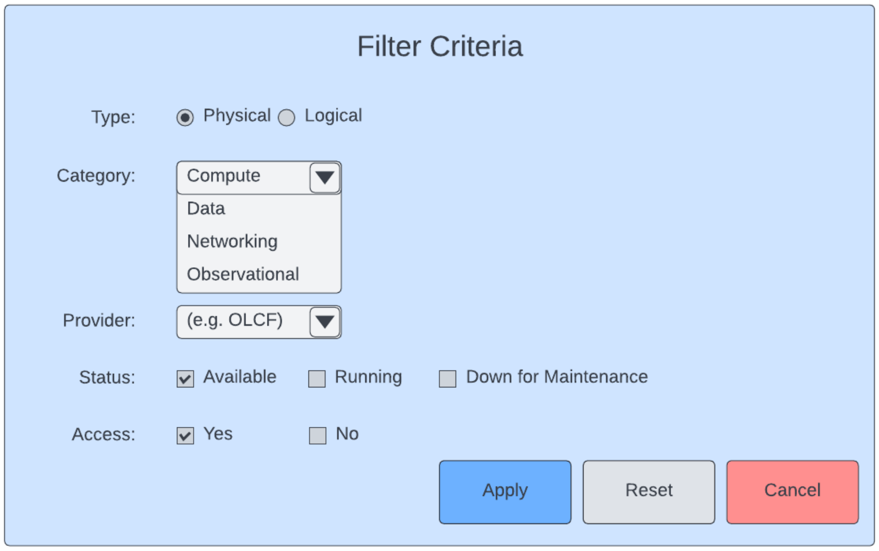
- Listing view for each Resource should show:
  - Title
  - Thumbnail image or icon if applicable / available
  - (if relevant) User has access - Yes / No
  - (if relevant) Status symbol / color / message (Available / Down / in use...)
  - Search bar to find Resource by keywords, type, ID, Owner, etc...
- Search and filter options
  - Search box to search by any details that describe Resources (title, keywords, etc.)
  - Filters for:
    - Physical / logical Resource
    - Category of Physical or Logical Resource
      - E.g. Physical resources - Compute, Data, Networking, Observational
    - Provider - Organization (E.g. OLCF)
    - Status - whether the Resource is Available, Down for maintenance, etc.
    - Access - whether this User (and implicitely the projects they are part of) can access each of the displayed Resources

An example interface is depicted in Figure A.10 and Figure A.11



Most Common Resources	Type	Category	Provider	User Access	Status
Summit	Physical	Data	OLCF	Yes	In Use
Frontier	Physical	Data	OLCF	No	Running
Miller (Hall A)	Physical	Networking	NCCS	--	--
Ninja	Physical	Data	NCCS	Yes	Running
Microscope	Physical	Observational	ORNL	Yes	In Use

Figure A.10. A list of the most common resources is shown.



Type: [✓ Physical] Logical

Category: Compute / Data / Networking / Observational

Provider: (e.g. OLCF)

Status: [✓ Available] Running Down for Maintenance

Access: [✓ Yes] No

Figure A.11. Filters can be applied to specify a search.

Detailed View of a Resource

Preconditions

User is registered and logged into INTERSECT

Postconditions

User can see a detailed view of their physical or logical resource

Methodologies

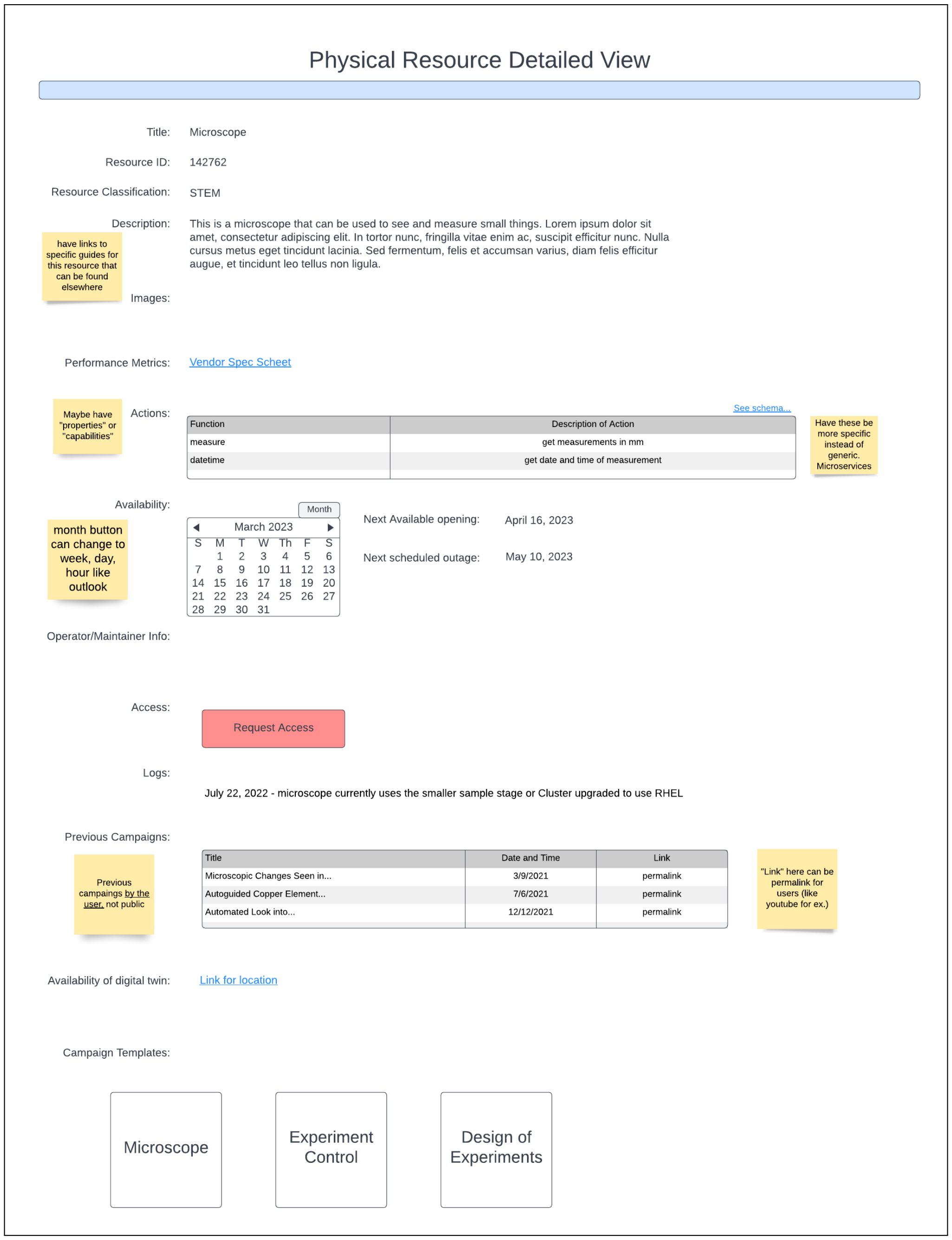
- Physical resources - instruments, compute, data, network
- Properties shown in listing
  - Title (abbreviate if necessary)
  - Thumbnail image (if available)
  - Current status
- Detailed view:
  - Title
  - Resource ID
  - Resource Classification. E.g. - Physical.Microscope.STEM
  - Description with:
    - Capabilities - this will invariably point to the modularity of the resource, allowing components to be swapped out to suit Campaigns better.
    - Examples of when and when not to use it
    - Examples of how to provide parameters if applicable
  - Thumbnail image
  - Images - of the different instrument setups
  - Performance metrics - or link to vendor's spec sheet
  - Actions -
    - listing of key actions one can perform with this resource very similar to any existing API documentation
      - Function name
      - Description of action
    - Link to full list of supported actions / documentation page
  - Availability -
    - Current: Use same colors and symbols as listed above.
    - Future: show calendar w/ lead time (to setup) for instruments
  - Operator / Maintainer information
    - Setup / tear-down times
    - Operator / Maintainer? What if there are several - the User doesn't know who would be assigned. Why
  - Access:
    - If user doesn't have access, the User clicks on a button to request access from the Owner -
      - present a button to request access.
      - The button points to a popup window with a few fields:
        1. Proposal number
        2. Proposal date (optional)

            3. Notes - any notes that would help the Owner validated the User, proposal number against internal processes for their Organization
    * If user does have access - show one of two statuses:
        · Under review - yellow / orange
        · Available - green check mark
    * Challenge:
        · INTERSECT will assume that, in reality / practically speaking, Organizations / Owners may already have a proposal process to grant access to their Resource(s) and that these processes do not have open APIs to access:
            1. components of the proposal (intent, needs, methods, duration…)
            2. list of pending proposals
            3. state of proposals
            4. whether or not a user has been granted access
        · Also, the nature of these proposals will certainly vary wildly between Organizations, so it may be overly tedious / pointless to reconcile / mirror / synchronize the INTERSECT interface and organization's proposal interfaces.
        · The processes to review and approve access also vary wildly. For example, CNMS's application and review process is entirely over emails. OLCF has a more streamlined system on RATS.
        · Furthermore, the process to grant access to a User involves reviewers, etc. who are not part of INTERSECT
        · The Owner of this Resource in INTERSECT may not be the same person who runs the "Resource Utilization Council" for the Organization.
        · Thus, it is safe to assume that INTERSECT will be out-of-sync with the external proposal / access management processes.
        · Therefore, INTERSECT could ask the User to provide the approved confirmation number / proposal number that the Owner of the Resource could validate within their internal process and allow this User to access the Resource. This allows Owners to incorporate a workflow that suits their Organization / Resource to include mandatory training / certification before the User is authorized to independently use the Resource.
  - Logs
    * Relevant changes to the resource. E.g. - microscope currently uses the smaller sample stage or Cluster upgraded to use RHEL ?.?
  - Listing of past Campaigns / campaigns that this user used this resource:
    * Title
    * Date and time
    * Link
  - Availability of digital twin with link
    1. list caveats and other assumptions
  - Campaign template(s):
    * Listing with abbreviated description with link to template
  - Reviews and ratings?
- Logical resources:
  - These can be divided into two categories:
    * Custom / Optimized - these are software applications that:

        - are computationally intensive
        - that are tailored for specific underlying Physical Resource(s), typically for time-sensitive / time-critical applications.
        - may be highly domain-specific
        - are in the form of a software container or a job script on a compute cluster.
        - Examples include:
          1. digital twins for physical resources or phenomena,
          2. job scripts for simulations that are optimized for one or more specific compute resources
          3. deep learning training job optimized for a specific compute resource or architecture (e.g. to take advantage of fixed precision available only on high-end GPUs, NCCL, InfiniBand, etc.)
          4. large-scale multi-variate statistical analysis code (e.g. pbdR)
      * Versatile - software applications that:
        - are not computationally intensive
        - will not be used for time-sensitive or rate limiting for a real-time Campaign
        - can run on most compute resources, especially the infrastructure that supports INTERSECT's web interface
        - are specified via containers or functions written in popular languages such as Python or R
        - Examples include:
          1. Simple data visualization
          2. Simple data analytics / statistics
          3. Lightweight inference on pretrained machine learning or deep learning models
  - Properties to show in listing:
      * Title (abbreviate if necessary)
      * Resource ID
      * Thumbnail image (if available)
      * Version number or (better yet:) Last update - when the twin was verified to match with the physical counterpart. This assumes that the physical counterpart does evolve with time and the virtual resource
  - Detailed view:
      * Title
      * Resource ID
      * Resource Classification - E.g. - Logical.Analysis.Visualization.2DImage
      * Thumbnail image
      * Images: Especially for visualization Resources; simulation applications could use this space to illustrate benchmarking results, etc.
      * Description
        - Capabilities
        - Comparison with one or more physical counterparts for Digital Twins
        - Assumptions / caveats
        - Turnaround time / response rate?
      * Actions:

        - Listing of key actions that can be performed with this Resource - a subset of the Resource's API
          1. Function name
          2. Description of action
        - Link to full listing of all supported actions / documentation
    - Availability:
        - Almost always "Available".
        - "Down" when Resource is being investigated for bugs or incompatibilities due to changes in dependencies.
    - Access - Almost always "Available" unless licensing restrictions limit access for some reason.
    - Computational resources that can support this Resource - primarily for Custom / Optimized Virtual Resources
        - (Compute) Resource name
        - Supported software version
        - Performance metrics (e.g. response time) or point to existing papers with benchmarking information
        - , availability, and "add" button?
          1. "CNMS NVIDIA DGX 2" `''Available''` "Add"
          2. "OLCF Frontier" `''Not Available''` <>
        - Given that the users will not be setting up the application themselves, it may not be necessary to show how the twin should be accessed (e.g. module or container) and set up.
    - Required computational resources (not required since we allow apps to only run on certain resources in an automated manner):
        - Architecture: (e.g. "Requires x86 architecture. Does not work on POWERPC")
        - CPU: recommend / required CPU cores
        - Host memory: recommended / minimum requirement
        - Acceleration: (e.g. "requires NVIDIA Volta V100 32GB or higher. Does not work on AMD GPUs")
        - Cluster size: (e.g. "requires 16 nodes with X cores and
        - Storage: (e.g. "100 GB per simulation with these nominal parameters")
        - Other: (e.g. "Needs InfiniBand, NVLink, half precision, etc.)
    - Logs
        - Relevant version changes listing bug fixes, etc.
    - Listing of Campaigns that used this resource in the past
        - Title
        - Date and time
        - Link
    - Related Resources - Relevant for:
        - Digital Twins
        - Simulation applications that are related to other Physical Resources
    - Reviews and ratings?
  - Button to add a new Resource into INTERSECT

An example interface is depicted in Figure A.12, Figure A.13, and Figure A.14



Physical Resource Detailed View

Title:	Microscope
Resource ID:	142762
Resource Classification:	STEM
Description:	This is a microscope that can be used to see and measure small things. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In tortor nunc, fringilla vitae enim ac, suscipit efficitur nunc. Nulla cursus metus eget tincidunt lacinia. Sed fermentum, felis et accumsan varius, diam felis efficitur augue, et tincidunt leo tellus non ligula.
Images:	
Performance Metrics:	Vendor Spec Scheet

have links to specific guides for this resource that can be found elsewhere

Actions:

Maybe have "properties" or "capabilities"

See schema...

Function	Description of Action
measure	get measurements in mm
datetime	get date and time of measurement

Have these be more specific instead of generic. Microservices

Availability:

month button can change to week, day, hour like outlook

Month

March 2023

S	M	T	W	Th	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Next Available opening: April 16, 2023

Next scheduled outage: May 10, 2023

Operator/Maintainer Info:

Access:

Request Access

Logs:

July 22, 2022 - microscope currently uses the smaller sample stage or Cluster upgraded to use RHEL

Previous Campaigns:

Previous campaings by the user, not public

Title	Date and Time	Link
Microscopic Changes Seen in...	3/9/2021	permalink
Autoguided Copper Element...	7/6/2021	permalink
Automated Look into...	12/12/2021	permalink

"Link" here can be permalink for users (like youtube for ex.)

Availability of digital twin: Link for location

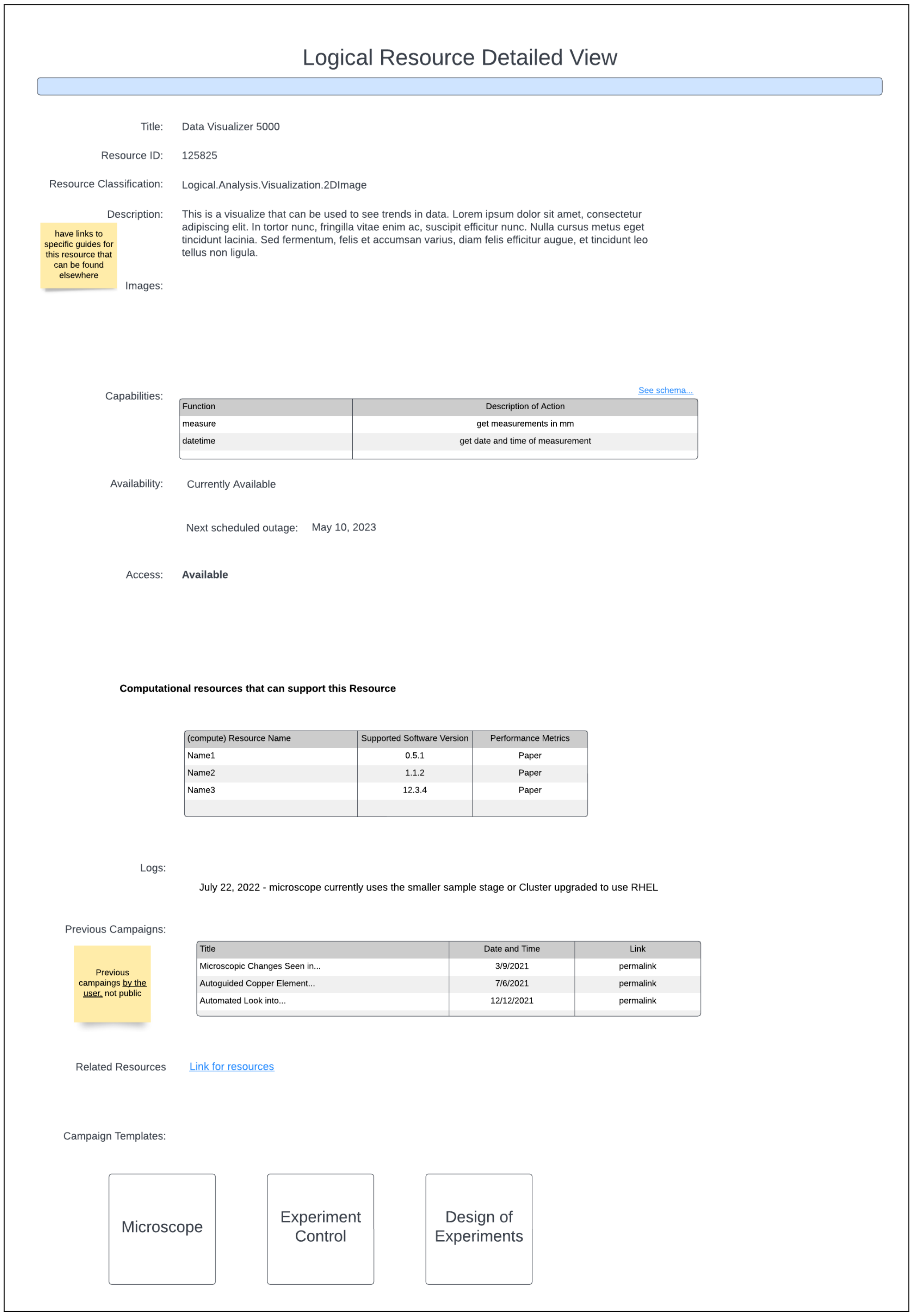
Campaign Templates:

Microscope

Experiment Control

Design of Experiments

Figure A.12. Detailed view of a physical resource.



Logical Resource Detailed View

Title:	Data Visualizer 5000
Resource ID:	125825
Resource Classification:	Logical.Analysis.Visualization.2DImage
Description:	This is a visualize that can be used to see trends in data. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In tortor nunc, fringilla vitae enim ac, suscipit efficitur nunc. Nulla cursus metus eget tincidunt lacinia. Sed fermentum, felis et accumsan varius, diam felis efficitur augue, et tincidunt leo tellus non ligula.

have links to specific guides for this resource that can be found elsewhere

Images:

Capabilities:

See schema...

Function	Description of Action
measure	get measurements in mm
datetime	get date and time of measurement

Availability: Currently Available

Next scheduled outage: May 10, 2023

Access: **Available**

Computational resources that can support this Resource

(compute) Resource Name	Supported Software Version	Performance Metrics
Name1	0.5.1	Paper
Name2	1.1.2	Paper
Name3	12.3.4	Paper

Logs:

July 22, 2022 - microscope currently uses the smaller sample stage or Cluster upgraded to use RHEL

Previous Campaigns:

Previous campaings by the user, not public

Title	Date and Time	Link
Microscopic Changes Seen in...	3/9/2021	permalink
Autoguided Copper Element...	7/6/2021	permalink
Automated Look into...	12/12/2021	permalink

Related Resources Link for resources

Campaign Templates:

Figure A.13. Detailed view of a logical resource.

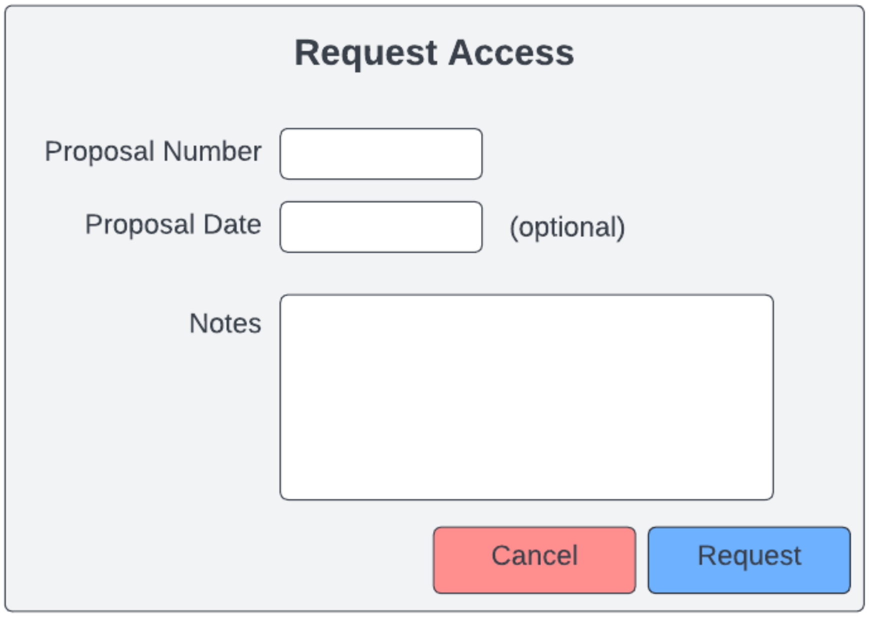


Figure A.14. Prompt to request access to a resource for a campaign.

User Notification Panel

Preconditions

The User is registered and logged in to INTERSECT

Postconditions

The user is able to see a notification panel for scheduled, running, and recent campaigns as well as the resources to which they have access.

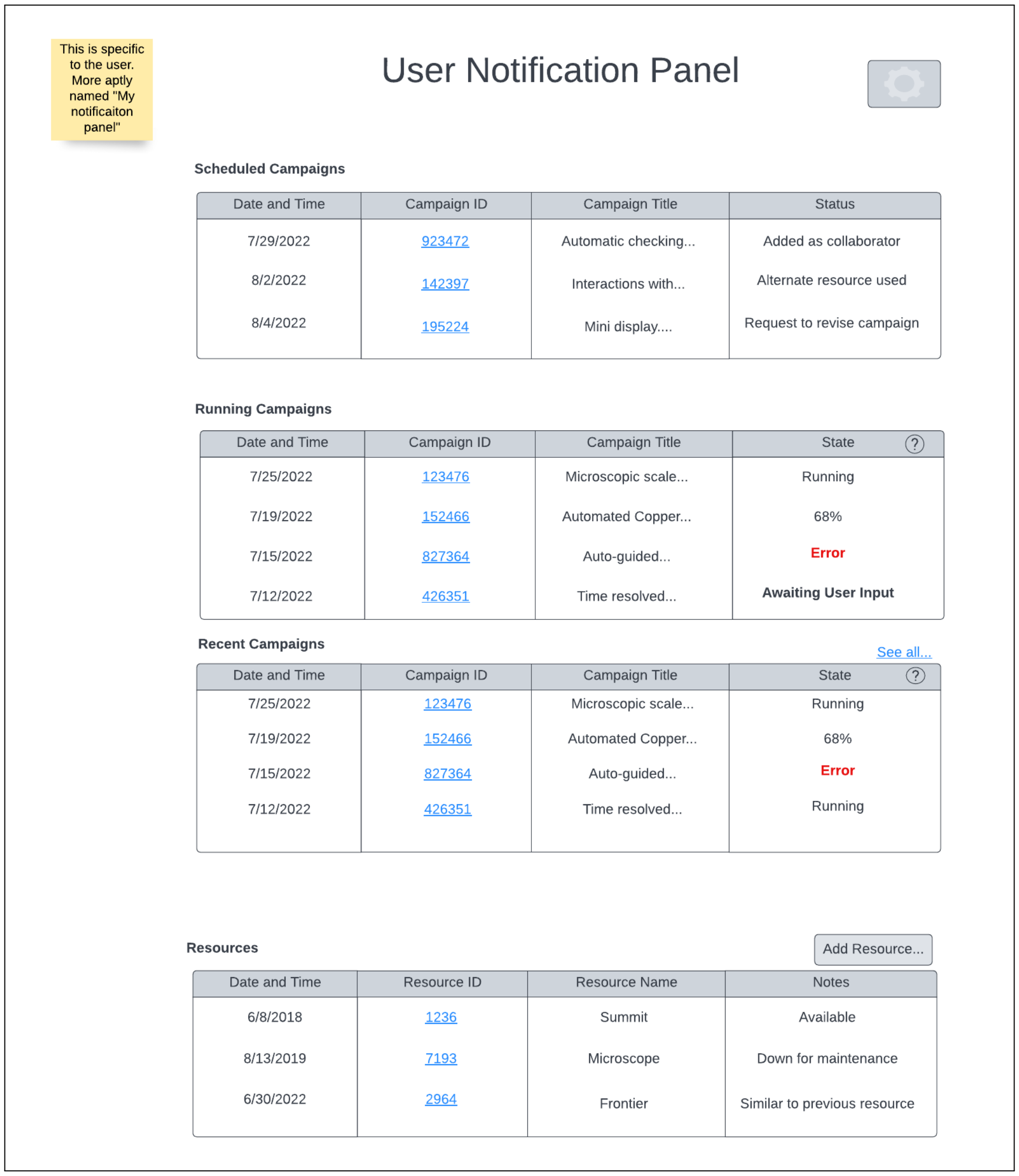
Methodologies

- The Notifications Panel shown in Figure A.15 would be visible only to this User.
- It could have different sections providing tidbits on status updates / actions to take:
  - Campaigns
    - Running Campaigns - Listing view with following columns:
      - Date and time
      - Campaign ID -
        1. For running Campaigns - link takes the User directly to the live dashboard for the Campaign
        2. For Campaigns that failed - link that takes the User to the Detailed view of the Campaign
      - Campaign title (abbreviated)
      - State
        1. For running Campaigns - display progress this is obvious or just display "Running"
        2. For Campaigns that failed - just display "Error" in red to grab attention
      - ~~Estimated time to completion - sometimes this is not obvious so this is a field we will ignore for simplicity~~
    - Scheduled Campaigns - Listing view with following columns:
      - Date and time
      - Campaign ID - link that takes the User to the Detailed view of the Campaign
      - Campaign title (abbreviated)
      - Status messages for situations:
        1. Added as a collaborator on another User's Campaign
        2. moving forward because one or more Operators completed setup of their Resources
        3. cancelled because one or more critical steps (PI rejected / Resource was unavailable and no alternatives were available) were impossible to complete
        4. Alternate Resource used (by Owner / Operator) in place of requested Resource
        5. Request to revise Campaign configuration in order to move Campaign forward
  - Resources - Listing view with following columns:
    - Date and Time
    - Resource Name
    - Resource ID
    - Note. Examples include:
      - Resource (frequently used by this User / one this user has access to) taken down for maintenance

- Resource (frequently used by this User / one this user has access to) that was down is now available for use
- new Resource similar to those used by User added to INTERSECT

1. This can potentially become annoying. User should have the ability to customize notifications

An example interface is depicted in Figure A.15


This is specific to the user. More aptly named "My notificaiton panel"

User Notification Panel

Scheduled Campaigns

Date and Time	Campaign ID	Campaign Title	Status
7/29/2022	923472	Automatic checking...	Added as collaborator
8/2/2022	142397	Interactions with...	Alternate resource used
8/4/2022	195224	Mini display....	Request to revise campaign

Running Campaigns

Date and Time	Campaign ID	Campaign Title	State ?
7/25/2022	123476	Microscopic scale...	Running
7/19/2022	152466	Automated Copper...	68%
7/15/2022	827364	Auto-guided...	**Error**
7/12/2022	426351	Time resolved...	**Awaiting User Input**

Recent Campaigns

See all...

Date and Time	Campaign ID	Campaign Title	State ?
7/25/2022	123476	Microscopic scale...	Running
7/19/2022	152466	Automated Copper...	68%
7/15/2022	827364	Auto-guided...	**Error**
7/12/2022	426351	Time resolved...	Running

Resources

Date and Time	Resource ID	Resource Name	Notes
6/8/2018	1236	Summit	Available
8/13/2019	7193	Microscope	Down for maintenance
6/30/2022	2964	Frontier	Similar to previous resource

Figure A.15. Notification panel is specific to each user.

Notification Customization by User

Preconditions

The User is logged in to INTERSECT and is viewing the Notification Panel.

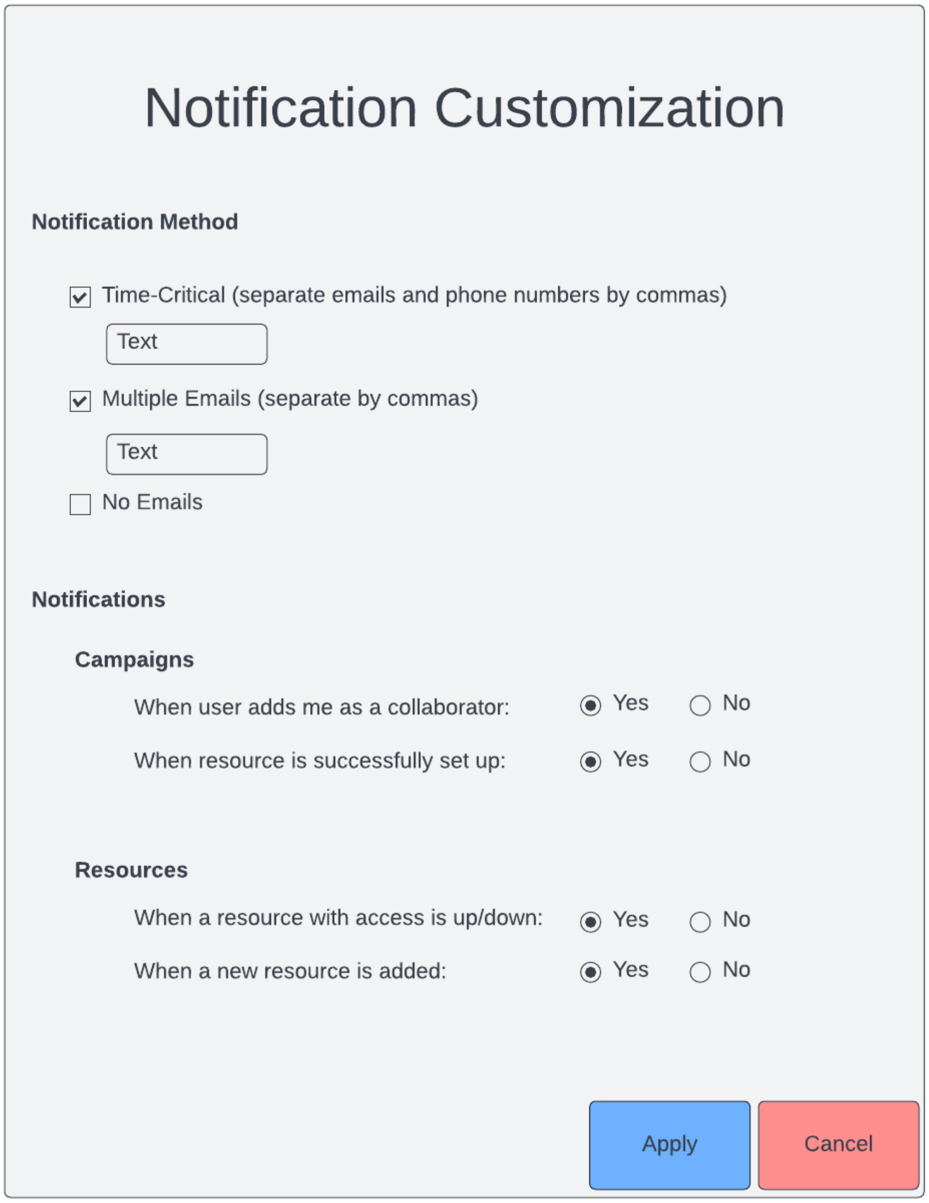
Postconditions

The User is able to edit which notifications they get and how they receive them.

Methodologies

- The Notifications Panel would have an options button that would reveal a settings menu for notifications. This panel would have the following items:
  - Notification method -
    - Time-critical : options to specify one or more email addresses and phone numbers for SMS
    - Other:
      - No email
      - options to specify one or more email address
  - Notifications
    - Campaigns
      - Running
      - Scheduled -
        1. When User adds me as a collaborator - Yes / No
        2. When Resource is successfully set up - Yes / No
    - Resource
      - Up / Down for Resources with access - Yes / No
      - New Resource - Yes / No

An example interface is depicted in Figure A.16



Notification Customization

Notification Method

[✓] Time-Critical (separate emails and phone numbers by commas)

Text

[✓] Multiple Emails (separate by commas)

Text

[ ] No Emails

Notifications

Campaigns

When user adds me as a collaborator:	[✓ Yes]	No
When resource is successfully set up:	[✓ Yes]	No

Resources

When a resource with access is up/down:	[✓ Yes]	No
When a new resource is added:	[✓ Yes]	No

Figure A.16. Users can choose which notifications to get and how to get them.

User Composes and Schedules a Campaign

Preconditions

The user is registered and logged in to INTERSECT and a specific subtype has been assigned to them.

Postconditions

The user is able to compose a campaign after choosing either a blank template or one from the gallery, filling out the appropriate contextual fields, and configuring the run-time behavior. The campaign is submitted to the orchestration system.

Methodologies

User composes a Campaign with desired components. The composition can happen in two different ways:

- Blank template - empty canvas
  - In an ideal world, campaign composition could be akin to setting up a workflow using a GUI or using a LabVIEW "virtual instrument" composition interface.
    - Users drag and drop blocks onto a canvas and connect them each other using virtual wires
    - Blocks would be:
      1. Actions enabled by Resources:
         (a) Physical resources such as:
            - Instruments (e.g. microscope.takeImage)
            - compute resources - shell script / job script specified here
            - data resources - typically data transfer commands if they are not automatically handled by INTERSECT when blocks are connected.

         (b) Virtual resources:
            - Custom
              - o Forward - Simulation modules
              - o Inverse - data analytics, fitting to physical models, etc.
            - Versatile
              - o Visualization widgets for the live-monitoring dashboard
      2. Basic programming and logical constructs such as loops, conditionals, comparators, etc.
    - Blocks would almost always require inputs and produce outputs:
      - Inputs such as:
        1. Configuration parameters for the resource / algorithm
        2. ~~Which resource to run on~~
           (a) ~~Hovering over or clicking shows a small list of recommended resources with a color signifying their applicability rating~~
           (b) ~~The applicability rating could come from a variety of factors including:~~
              - o ~~Timeliness of result~~
              - o ~~Reliability (e.g. for network)~~
              - o ~~Cost~~
              - o ~~Accuracy~~
              - o ~~Precision~~
        3. Alternate / redundant resources for fail-safe - this typically applies to compute resources that can be swapped out without affecting the workflow

            4. Data stream or file(s)
          · Outputs
            1. Any data stream or files that are produced
      * Realistically, the campaign composition could use existing open workflow schemas. In other words, the workflow would be specified in some kind of a markup or markdown language.
          · However, this may require the Users to be educated on how to compose workflows.
          · It might be helpful to be able to graphically (static) represent the workflow for Users to be able to quickly understand the workflow at hand

- Template from the gallery or Past Campaign - completely filled out
  - User is shown a composition (graphical or otherwise) of the workflow.
  - User edits the workflow to suit their needs. This modification may be done via a graphical or script-based interface (whichever is implemented).
  - User has access to the same capabilities and functionalities as if they were starting from the Blank Template
- Regardless,
  - User is asked to fill out or edit prepopulated metadata / contextual fields:
    * Title
    * Intent
    * (Scientific) Background
    * Description (of how the intent is accomplished with this workflow)

User configures run-time behavior

- “Offline” monitoring mechanisms (email, SMS, ...)
  - Options need to be presented on exactly what information is sent via emails / SMS transmission.
    * Overall campaign progress
    * Overall health of the resources
    * Key figures that illustrate the current state / progress of the campaign
- “Online” monitoring via Dashboard widgets
  - Progress bars (one for the overall campaign of Campaigns, and another for the current Campaign (if relevant))
  - Timers - time elapsed, time remaining
  - Meters or LEDs for Infrastructure health like the “health” of the primary resources, network bandwidth,
  - Performance metrics specific to the campaign that are more relevant to the user such as “frames/sec”, or “Samples/hour”, etc.
  - Real-time Image, spectral, volumetric, phase diagram, and other scientific plots that update in real-time as and when fresh data is made available
    * If relevant and technically possible, it would be nice to have the option for the user to go back in time to revisit previous trends
  - Button to ask for help from the Operator / Maintainer
  - Button to abort Campaign

User submits campaign

- User is taken to the campaign save page
- User is told that campaign has been saved with a unique ID
- User is asked if they want to add the template to the catalog
  - This could be triggered if there aren't any existing templates that use the main experimental and computational resources.
  - If it is triggered, it goes through a review process
  - For example, the catalog only contains templates of a microscope being used with a DGX box for real-time steering. The current template uses a similar microscope with an FPGA for feedback and Summit for building a virtual model of the material under the microscope. In such cases, the review process would find this template to be substantially different than the templates that already exist. Upon successful completion of the Campaign, the Operator may then approve this request and add this as a template. The Operator may ask the User to flesh out details regarding the template
- User is informed they would need to authenticate and authorize the following resources prior to the start of the Campaign

User schedules Campaign

- User is shown possible slots for running the Campaign.
  - The slots are determined primarily / largely by the observational resources that can / need to be scheduled several hours or days in advance.
  - Computational and data resource availability / down-time will also be taken into account but in generally may be assumed to be available unless the Campaign is being scheduled to start in the next few minutes
- User picks one slot to run the Campaign
- Campaign shows up on "upcoming" / "scheduled Campaigns" with expected time.
- Clicking on this allows user to resume session.

An example interface is depicted in Figure A.17 and Figure A.18

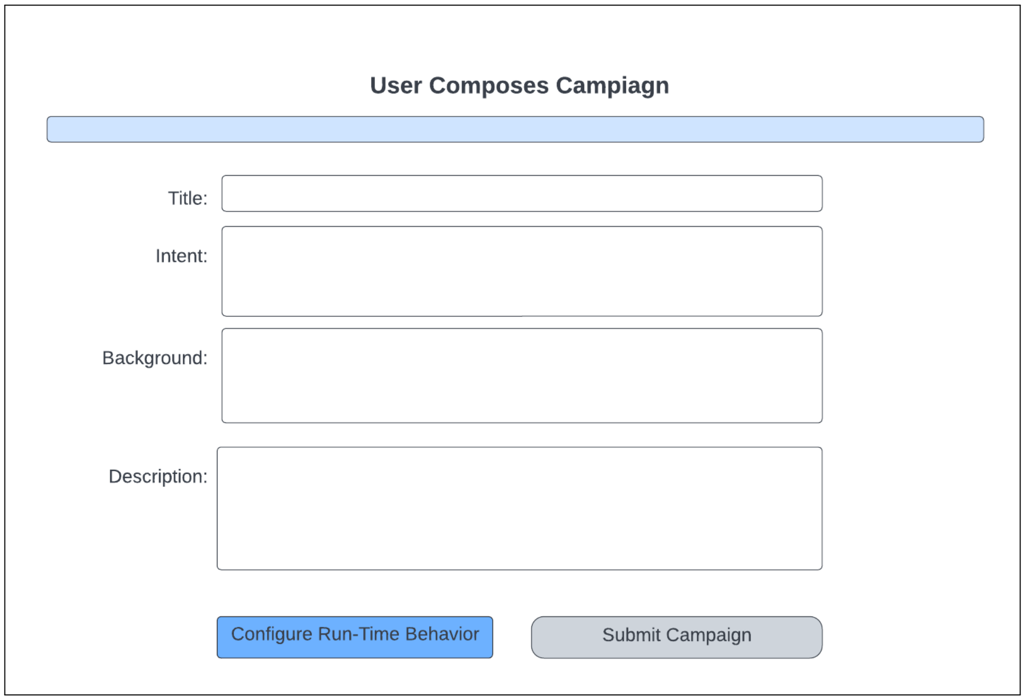
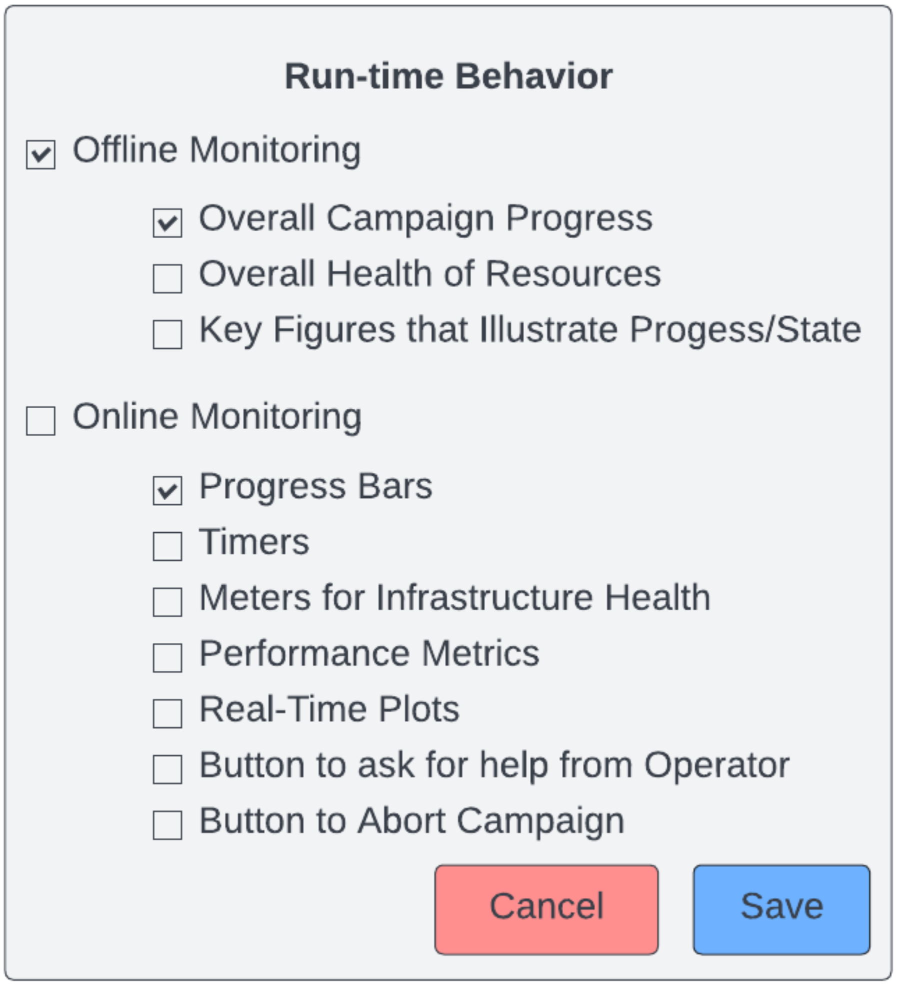


Figure A.17. These fields are prompted regardless of chosen template.


Run-time Behavior

- [x] Offline Monitoring
  - [x] Overall Campaign Progress
  - [ ] Overall Health of Resources
  - [ ] Key Figures that Illustrate Progess/State
- [ ] Online Monitoring
  - [x] Progress Bars
  - [ ] Timers
  - [ ] Meters for Infrastructure Health
  - [ ] Performance Metrics
  - [ ] Real-Time Plots
  - [ ] Button to ask for help from Operator
  - [ ] Button to Abort Campaign

Figure A.18. User must configure run-time behavior before starting campaign.

Start Campaign

Preconditions

The User is logged into INTERSECT and has been notified that the scheduled campaign in ready.

Postconditions

The user has authenticated the resource(s) needed in order to start the campaign and the user can see the running campaign dashboard.

Methodologies

- User is notified via email about INTERSECT being ready for the Campaign and is reminded 10-15 minutes before the actual start time
- User logs back into INTERSECT
- User views list of pending Campaigns from the homepage
- User selects appropriate Campaign (perhaps this surfaces up by itself on the home screen so the previous steps can be avoided)
- User elects to start the Campaign when the time to start the Campaign has arrived
- User is shown a list of resources that require them to authenticate / authorize
  - User is taken to the specific resource's page to authenticate / authorize
  - This process completes until all resources in the list are fully authenticated /authorized
  - The status needs to be pulled from the resource as opposed to whether or not the User went through INTERSECT to complete the process
  - User views auth tokens / certificates for resources and renews
- User is taken to the running Campaign dashboard view
- User monitors state of running Campaign passively
  - User may be notified out-of-band (email / SMS) according to their configuration
  - User views the live dashboard if they choose to

An example interface is depicted in Figure A.19

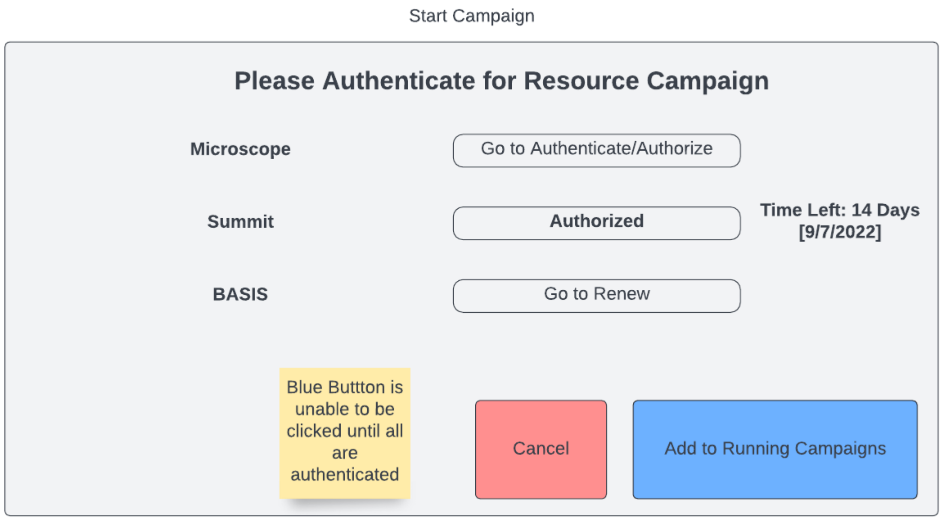

Start Campaign
Please Authenticate for Resource Campaign
Microscope
Go to Authenticate/Authorize
Summit
Authorized
Time Left: 14 Days [9/7/2022]
BASIS
Go to Renew
Blue Buttton is unable to be clicked until all are authenticated
Cancel
Add to Running Campaigns

Figure A.19. User authenticates resource for the campaign.

Steer Campaign

- Option 1 - autonomous / automated steering of Campaign or campaign using ML/DL/BO
  - User has no control and only monitors the Campaign passively via the dashboard
- Option 2 - User steers Campaign manually from time to time using one or more widgets.
  - This may involve:
    - modifying / overriding parameters
    - handling warnings
- Option 3 - User contacts

End Campaign

Preconditions

The user is logged into INTERSECT and is currently running a campaign.

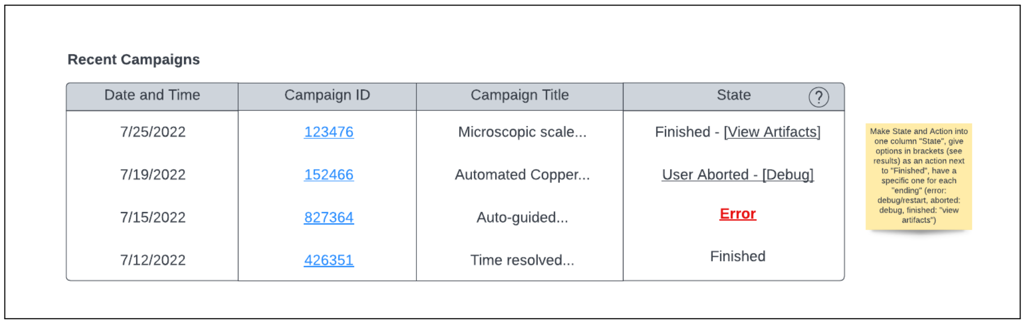
Postconditions

The campaign has come to an end and the user can see it listed under “Recent Campaigns”. They can click on the links to see the details of the campaign ending.

Methodologies

- Option 1 – Natural conclusion
  - Campaign ends by itself based on the termination condition
- Option 2 – User aborts Campaign
  - User clicks on “Stop” or similar button on dashboard to end Campaign prematurely
- Option 3 – Error in Campaign
  - One or more components decide to end the Campaign earlier due to an error that the Campaign cannot recover from.
- Dashboard elements that allow steering or restarting of certain components of the workflow are disabled.
- Dashboard may continue to be accessible for the user to scrub through the timeline and replay the Campaign using telemetry / saved components of the data stream.
- User receives an email notification about the end of the Campaign with a link to view details regarding the completed Campaign.
- This Campaign is moved from “Ongoing” to “Past” Campaigns

An example interface is depicted in Figure A.20



Recent Campaigns

Date and Time	Campaign ID	Campaign Title	State
7/25/2022	123476	Microscopic scale...	Finished - [View Artifacts]
7/19/2022	152466	Automated Copper...	User Aborted - [Debug]
7/15/2022	827364	Auto-guided...	Error
7/12/2022	426351	Time resolved...	Finished

Make State and Action into one column "State", give options in brackets (see results) as an action next to "Finished", have a specific one for each "ending" (error: debug/restart, aborted: debug, finished: "view artifacts")

Figure A.20. A dashboard of recent compaigns is shown at the end of the campaign.

User Requests to Add a New Resource to INTERSECT

Preconditions

User is registered and logged into INTERSECT

Postconditions

If the request to add the Resource were approved, the User would now be the Owner of this Resource

Methodologies

- Basic details:
  - Resource ID: Generated when the Owner starts the process of requesting a new Resource to be added to INTERSECT
  - Title - visible to all Users (public facing). How would a User differentiate seemingly identical Resources?
  - Alternate Title - visible to all but Users - answers the question - do we need another one for internal use? For example, what if there are 6 identical Resources? CNMS has 4 microscopes that are nearly identical and are internally differentiated by a suffix - “East”, “West”, “North”, and “South”.
  - Resource Classification - Physical.Microscope.ScanningTransmissionElectronMicroscope or Logical.Visualization.
  - Description - capabilities, assumptions, caveats, recommended usage
  - Thumbnail image
  - Vendor - E.g NVIDIA. Irrelevant to Logical Resources
  - Model Name - DGX II
  - Vendor Spec-sheet - link (relevant to physical resources)
  - Images if relevant (e.g. visualization widgets, scientific instruments)
  - Key performance numbers
- Capabilities / Modes
  - Context: A given Resource may have multiple Capabilities / Modes in which the Resource can be operated / used. For example - measurement modalities in a scanning probe microscope - imaging / spectroscopy / hyperspectral imaging....
  - List of unique capabilities / modes that this Resource is capable of supporting. Per Capability / Mode:
    - Title
    - Description
      - Include caveats / tradeoff in relevant performance metrics that are not obvious to the end User.
    - Setup lead time
      - Duration
      - Operator / Maintainer - A Mode / Capability can be sufficiently nuanced and differentiated that a single Operator / Maintainer may not be knowledgeable enough to handle other Modes / Capabilities of the same Resource.
        1. Editable list of Users (add/remove) - this would be a very simple way to manage privilege elevation from User to Operator / Maintainer. Note that privilege elevation is valid only for this Resource. This User would have the same capabilities as

            everyone else would for other Resources - i.e - this Operator / Maintainer should not be able to maintain / look into management of another Resource.
         2. If multiple Operators / Maintainers are assigned to this Mode / Capability - logic for INTERSECT to use to automatically pick the most appropriate Operator / Maintainer from the list of Operator / Maintainers. It could be as simple as User X always does the setup and User Y always does the tear-down
    * Tear-down lead time
      · Duration
      · Operator / Maintainer
         1. List
         2. Logic

- Delegates
  - Listing of other Owner / Operator / Maintainers who could make edits (typically low-level parameters that the Owner may not be intimately familiar with) to the configuration on behalf of this Owner.
  - This list of Users may overlap with the Operator / Maintainers
  - Perhaps the Owner only fill out the high-level information or just proof-read whatever the Operator / Moderators put together on behalf of the Owner.
- Access
  - Link to application form / page to how a user can gain access
  - ~~API call to RATS like system to allow INTERSECT to query if a user has access or not - this will be used for the User's view~~
  - ~~API call to RATS like system to retrieve list of Users who have requested access to this Resource and if possible any progress regarding this request. - This will be used to show the Owner who has requested access when the view the detailed view of a Resource~~
  - Proposal ID field to capture from User - this will be used by Owner to manually lookup proposal within Organization's access control system / processes and then approve / deny access of this Resource to said User
- Version history / Change log:
  - Relevant to both physical and software resources
- Availability of digital twin with link
  - list caveats and other assumptions
- Campaign template(s):
  - Listing with abbreviated description with link to recommended templates
- Hooks
  - Physical resources:
    * Availability (if relevant)
      · Configuration / API call to:
         1. query current state
         2. future availability
         3. Reserve all or part of resource
            (a) Parameters that need to be passed
         4. think of using something like SuperFacilityAPI
      · OR Exchange / Outlook calendar integration

        - Lead time specified in hours - for configuring the setup of the resource before every Campaign, if relevant (especially for observational resources)
      - INTERSECT Resource Control daemon address (if relevant)
      - Globus Endpoint UUID for data transfers (if relevant)
  - Software resources:
      - Per computational resource:
        - Resource link in INTERSECT
        - Resource Name
        - Shell / job script for launching the application
      - ~~minimum requirements to run this app (CPU architecture, GPU architecture, parallel configuration, dependency software stack, etc.)~~
- Links to other resources either as virtual twin to a physical resource, visualization module for streaming data, or analysis results, etc.
- Alternate resources upon unavailability - listing
  - If applicable, Owner provides a list of comparable resources that could be used in place of this resource if and when this resource is down.

An example interface is depicted in Figure A.21 and Figure A.22

Process: Login Web

Figure A.23 applies to this and following use cases.

The login process is responsible for receiving user login information, authenticating the user, and loading an associated user profile. Authentication is delegated to an Authentication Server that receives appropriate credentials and returns an acknowledgment upon valid user credentials or an error message given invalid user credentials. The process of loading a user profile is more complicated. A user profile is a collection of metadata that corresponds to remembered state and preferences for a particular user. Specifically, to INTERSECT, a user profile contains information about who the user is, what resources they care about, and collections of helpful logical recipes.

Drilling into resources, a user profile is a registry that contains connection criteria, file paths, etc. that are referenced using key-value criteria. This allows for the user to apply friendly names or aliases to resources much like a user creating custom names for web bookmarks. However, unlike URLs resources within the user profile may be more complicated such as they may cache connection criteria to databases and have saved queries to views within that database. Similarly, they could represent a file and a location within the file. Additionally, resources could be an index of user history analogous to a history within a UNIX bash console.

Drilling into the context of logical recipes, the user profile may record collections of templated activities that are commonly observed from the user's activities. For example, if a user is commonly copying files from one location to another, the system may record a "copy recipe" that ask much the same as a template. It basically says, I've seen you do this type of activity before if you need to do it again I'll help you. Consider a use case where a scientist has a multistep process within their history for setting up an experiment. Maybe the scientist must first copy a configuration file from a staging location to an instrument such as updating firmware settings. Then the scientist needs to move data to that local machine and allocate a compute node. Finally, the scientist needs to load the machine and execute a pipeline. Each of these might be independent line items within their history, but the system is smart enough to take collections of items within the history and build a directed acyclic graph from them. This graph is then represented as a template using the pre-existing state. The next time that a user wants to perform a similar operation, they can look in their templates section of their user profile and select a template that the system has learned from their history. Upon selecting this template, the directed acyclic graph is color-coded to illustrate when the state is preloaded example state or if it has been updated by the user. For example, the background of a directed acyclic graph node might be white for default state and green for updated state. The goal here is to aid the user in being methodical and helping avoid human error. Similarly, as the user uses the system more, the system learns more about how the user works and adapts so that the user can be ever more efficient in their activities.

Another aspect to a user profile is the means of sharing subsets between users. For example, if a user is doing something complicated such as installing Linux, it is very difficult and error-prone unless the explicit commands are shown. The ability to share these types of commands or scripts enables users to onboard and become effective quickly by bootstrapping off one another.

Preconditions:

- The URL https://intersect.ornl.gov/ must be live.
- User has a valid and active ORNL account.
- Authentication Server is live and reachable.
- User Profile Service is live (optional, but preferred).
- Federated Recipes is live (optional, but preferred).

Post-conditions:

- A user is verified against their username and password credentials.
- If the User Profile Service is active, the user will have access to all of their predefined state, such as history and bookmarks.
- If the Federated Recipes is live, the user will have access to a collection of templated directed acyclic graphs that are either standardized by a community or derived from an individual user's rolling history.

Process: Request Resources

When a user has successfully logged into the intersect system, they may request resources. Within this process, a resource may be thought of as either a location or a named process. For example, a URL and hash code may be thought of as a location resource type. In contrast, Python scrnnipts and named queries may be thought of as named process resource types. When a user requests a resource, such as "discover spectrophotometer instruments," the system finds a corresponding script to query an active registry of instrumentation returning the subset of instrumentation corresponding to spectrophotometers. The script may also generate custom views of the result sets such as graying out the set of spectrophotometers that are not operational or currently in use by someone else.

Preconditions:

- the user has successfully logged into the system
- the user has submitted a resource query
- a registry of resources has been compiled as a collection of NFTs and smart contracts describing the sharing state machine

Post-conditions:

- the resource manager has responded with a result set corresponding to the user's resource query
- the user has the option to select from the available resources based on a variety of filter options such as tag or entity type.

Process: Compile DAG

After a user has requested resources, resources may be paired with federated recipes or constructed in two new user-defined directed acyclic graphs (DAGs). These graph structures represent a flowchart that describes how data moves from logical component to logical component until the work is completed and data is stored in some destination. Great examples of this are NodeRed, GNU Radio, and Apache Airflow. The DAG data structure may be thought of as a symbolic representation of the work to be done. However, for actual work to be performed, this logical structure must be instantiated against physical components such as computational nodes and instrumentation. To compile a DAG, the symbolic representation is sent to an interpreter to build a job.

Preconditions:

- the user is logged into the system
- the user has performed a resource query
- optionally the user has queried an appropriate federated recipe and potentially adapted it for their particular use case
- optionally, the user has constructed a new DAG from scratch

Post-conditions:

- The DAG is passed to the interpreter to change the representation from a symbolic version to a job which has the effect of allocating the resources such that for some scheduled allotment of time all resources have been mutexed so that only the current user is capable of accessing them without a manual override.
- A smart contract has been compiled corresponding to the set of resources and the project has been billed for their use.

Process: Submit Job

After a DAG has been compiled into a schedulable job, the job may be submitted to an orchestrator. At this point, all configuration metadata is bundled with completely resolved resource entities (this is the difference between having placeholders or relative paths with unambiguous absolute paths or static values). Within the orchestrator, the job is queued in a registry to be submitted to the scheduler for execution. At this point, the user or an administrator has the capability of canceling the job. This is analogous to a printer queue where a job is submitted, and a user can open a corresponding device manager program to manually cancel the job.

Preconditions:

- The user is logged into the system
- There are no errors in the symbolic representation of the DAG
- All configuration metadata is present and bundled with the job

Post-conditions:

- All resource placeholders such as NFTs are replaced with connection criteria, URLs, port mappings, micro-services, etc.
- All configuration data has been bundled such as the creation of immutable docker volumes that may be attached to the job
- All absolute paths have been tested such as a ping to assert connectivity
- All corresponding trigger time has been associated with the job within the orchestrator

Notes:

- Any errors will dequeue the job and post notifications to both an administrator and the user
- Consistent failures on a particular resource will cause status changes to the NFT such as a reliability attribute

Process: Register Job

When a job's trigger time has been activated, the orchestrator triggers to register the job bundle with the scheduler. The job bundle is interpreted by the scheduler giving it initialization instructions such as the ability to copy data from some named location to a staging area or stream data. Additionally, resources are corralled and mutexed by changing their state within a corresponding smart contract essentially preventing them from double use or cross communications. At this point the user is unable to manually cancel a scheduled job, only an administrator can do so. The job sits pending in the scheduler until its workflow is triggered.

Preconditions:

- There are no errors in the symbolic DAG
- All resources have been pinged to assert their existence and functionality
- An appropriate timeslot has been allocated roughly and associated with a trigger time within the orchestrator
- All configuration data has been bundled and is considered immutable

Post-conditions:

- A resource bound job is sent to the scheduler when a trigger time has been tripped within the orchestrator. This throttles the amount of activity on the scheduler (load-balancing).
- The scheduler receives the resource bound job and assigns an internal trigger time to start the job and assigns a job duration specifying the maximum allowed run time before the job is killed by the scheduler.

Process: Trigger Workflow

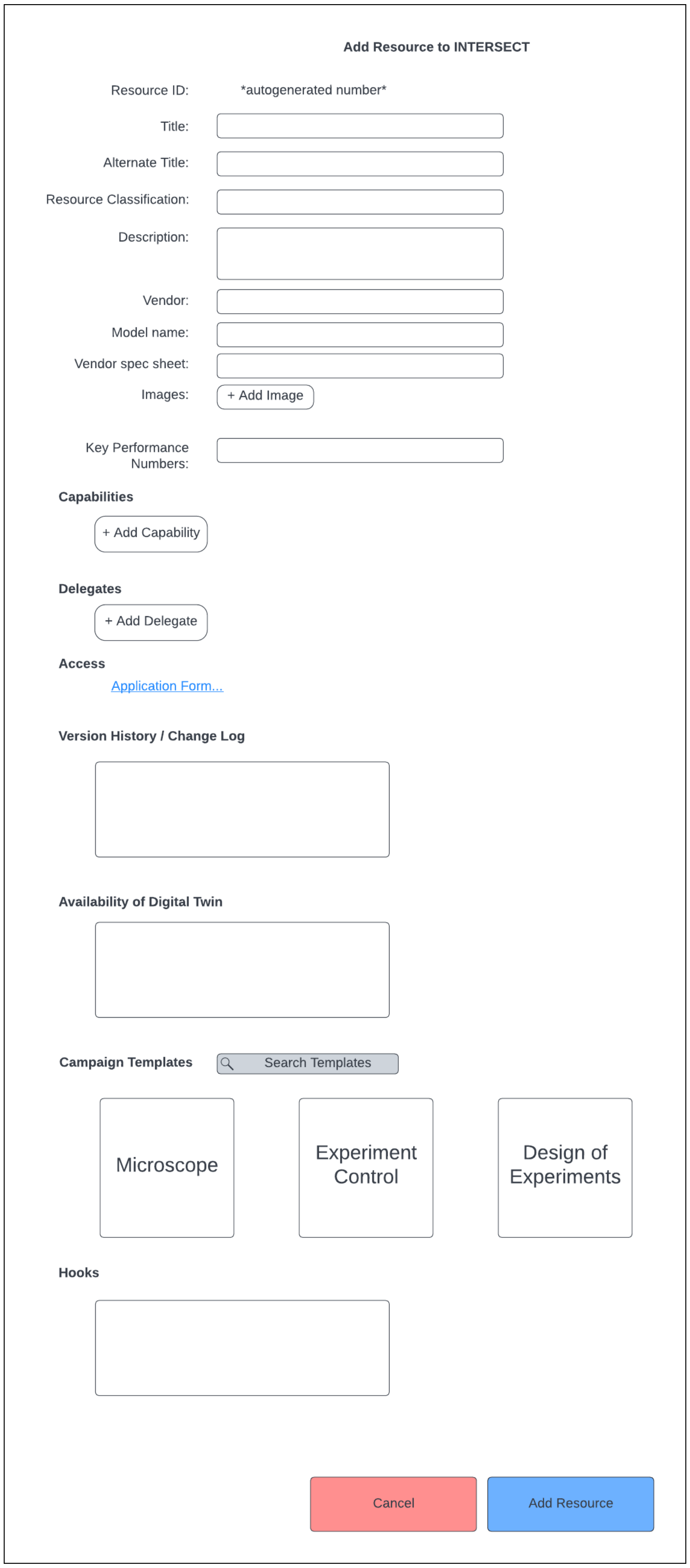
The scheduler is analogous to a game loop. A continually monitors a priority queue of incoming resource bundled jobs and active workflows. Its job is largely to decide when and where a workflow should run and if it should be killed. Within this loop, the scheduler is monitoring the incoming job queue. It pops the lead entry off the queue and places it in a dictionary keyed by trigger time. Each time the loop within the scheduler cycles, it evaluates the sorted list of keys to choose the next available slot in which to run a workflow. When the scheduler's clock time is greater than the minimum of the sorted keys, the minimum sorted key is queried from the dictionary to retrieve the corresponding resource bundled job. At this point, a new workflow is initialized with the corresponding resources. The new workflow is started and a pointer to the new workflow is added to the active workflows registry.

Preconditions:

- The scheduler has received a valid resource bound job on its incoming jobs queue
- All resources for overlapping periods of time are mutually exclusive

Post-conditions:

- The resource bundled job has been stored within the scheduler's incoming jobs registry to be available when the schedulers clock triggers the workflow start.
- The instantiated workflow is stored within the active workflows within the scheduler.
- Scheduler events have been associated with the newly created workflow so they may be reported



Add Resource to INTERSECT

Resource ID: *autogenerated number*

Title:

Alternate Title:

Resource Classification:

Description:

Vendor:

Model name:

Vendor spec sheet:

Images: + Add Image

Key Performance Numbers:

Capabilities

+ Add Capability

Delegates

+ Add Delegate

Access

Application Form...

Version History / Change Log

Availability of Digital Twin

Campaign Templates Search Templates

Microscope

Experiment Control

Design of Experiments

Hooks

Cancel

Add Resource

Figure A.21. Form to add resource to INTERSECT.

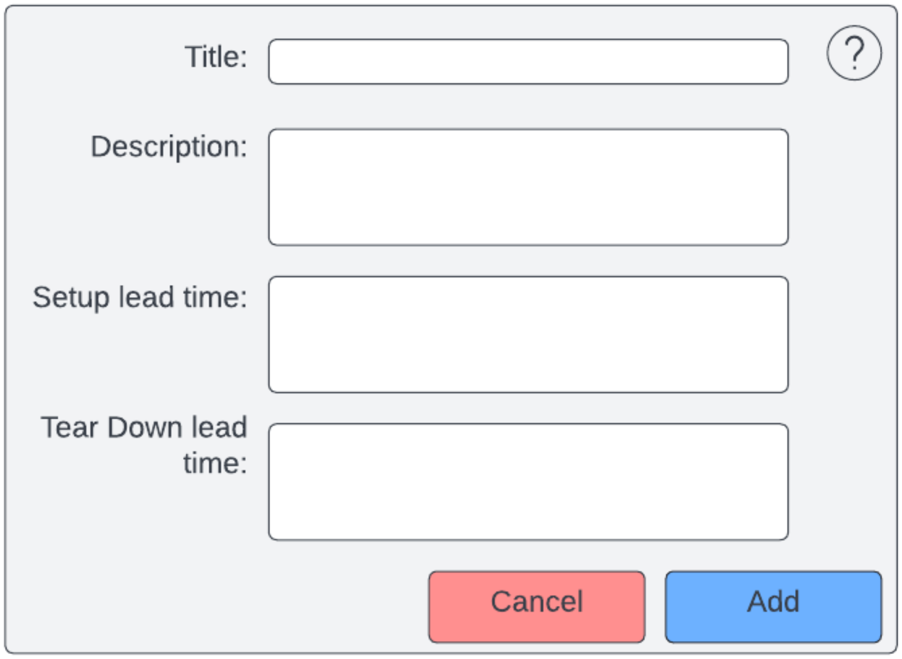


Figure A.22. Capability that resource is capable of supporting.

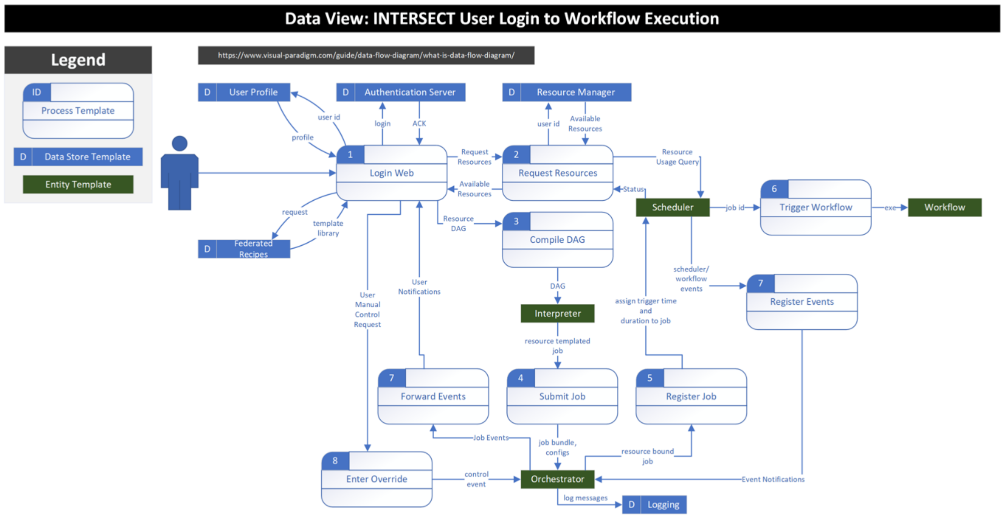


Figure A.23. Process perspective of logging into INTERSECT and building/executing a workflow.
This figure will be updated in a future version of this document.

A.2 OWNER

- Once logged into the Home screen in INTERSECT, an Owner would see a few more top-level views in addition to what the User sees

Owner Views Tasks and Responsibilities in the Notification Panel

Preconditions

The User is logged into INTERSECT with the owner role assigned.

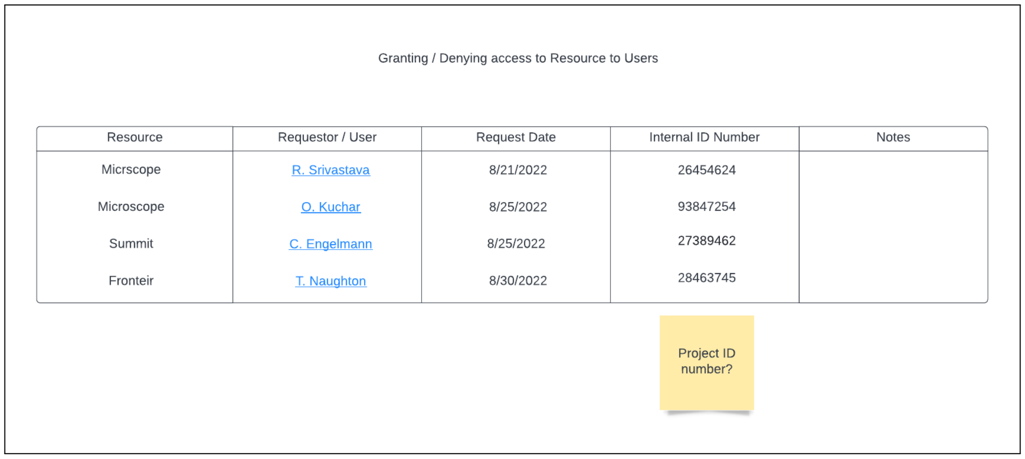
Postconditions

The Owner is able to see a resource dashboard where they can grant/deny access of resources to users.

Methodologies

- ~~Operator / Moderator assignment / triaging for Campaigns < —— ASSUME that INTERSECT does this automatically~~
  - ~~Succinct listing with following information (sort by date, resource, etc.):~~
    - ~~Resource~~
    - ~~Campaign title~~
    - ~~Campaign date~~
  - ~~Clicking on a Experiment takes the Owner to the detailed view about the Campaign where the Owner can~~
- Granting / denying access to Resource to Users
  - Succinct listing with the following information that can be sorted by resource or request date (urgency):
    - Resource
    - Requestor / User
    - Request date
    - Internal identification number
    - Notes
  - Clicking on a request takes the Owner to the User Management Panel

An example interface is depicted in Figure A.24



Granting / Denying access to Resource to Users

Resource	Requestor / User	Request Date	Internal ID Number	Notes
Micrscope	R. Srivastava	8/21/2022	26454624	
Microscope	O. Kuchar	8/25/2022	93847254	
Summit	C. Engelmann	8/25/2022	27389462	
Fronteir	T. Naughton	8/30/2022	28463745	

Project ID number?

Figure A.24. Owner’s view of resource notificaitons regarding access.

Owner manages permissions / roles using the User Management Panel

Preconditions

The user is logged into INTERSECT with the owner role assigned.

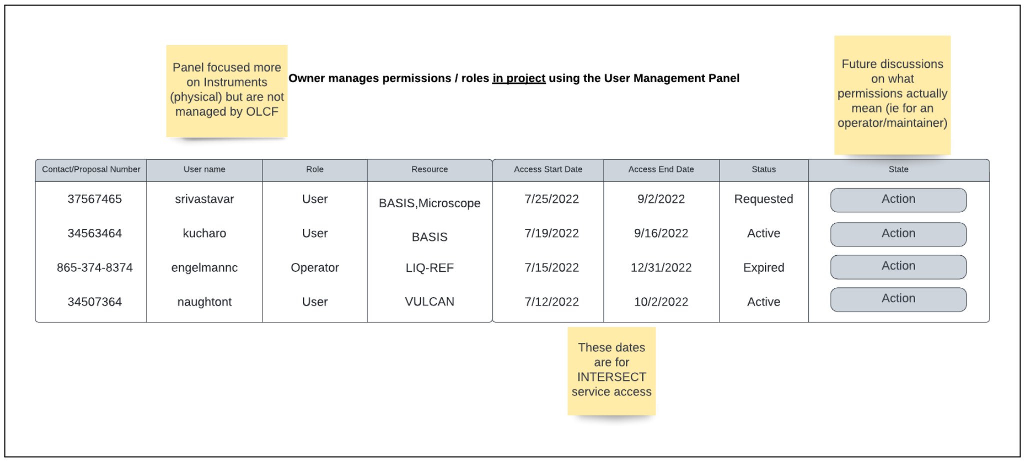
Postconditions

The owner is presented with a detailed dashboard that allows them to approve, renew, or revoke access of resources to users.

Methodologies

- Listing View:
  - Table that with following fields:
    - Contract number (for Operators / Maintainers) / Proposal number (for Users) - This will allow people to have access to a Resource for multiple discontinuous time periods
    - User name
    - Role - User / Operator / Maintainer
    - Resource(s) - use DB / UI trickery to keep table succinct instead of several rows for different combination of User Name x Resource
    - Access start date
    - Access end date
    - Status -
      - Requested / Pending
      - Active
      - Expired
    - Button(s) to:
      - Approve access
      - Renew access
      - Revoke access
  - The Listing View should allow the Owner to easily filter by Resource, User / Operator / Maintainer, Role, Date ranges, Status, etc.
  - The Listing View should allow the Owner to easily sort by any of the columns in the table

An example interface is depicted in Figure A.25 and Figure A.26


Panel focused more on Instruments (physical) but are not managed by OLCF
Owner manages permissions / roles in project using the User Management Panel
Future discussions on what permissions actually mean (ie for an operator/maintainer)
These dates are for INTERSECT service access

Contact/Proposal Number	User name	Role	Resource	Access Start Date	Access End Date	Status	State
37567465	srivastavar	User	BASIS,Microscope	7/25/2022	9/2/2022	Requested	Action
34563464	kucharo	User	BASIS	7/19/2022	9/16/2022	Active	Action
865-374-8374	engelmannc	Operator	LIQ-REF	7/15/2022	12/31/2022	Expired	Action
34507364	naughtont	User	VULCAN	7/12/2022	10/2/2022	Active	Action

Figure A.25. This management is done in project.

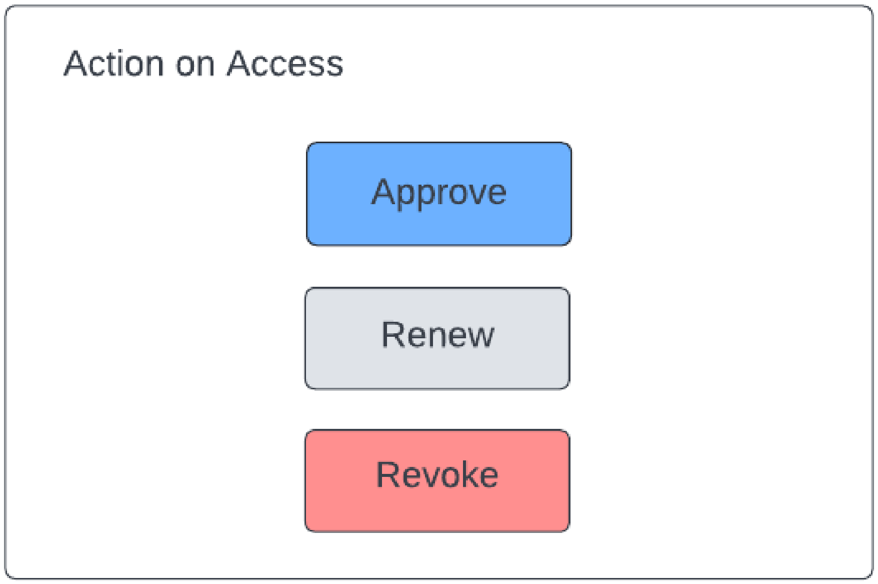


Figure A.26. Three options are given regarding access.

Owner monitors support levels and directions for Owners / Maintainers

Preconditions

The user is logged into INTERSECT with the owner role assigned.

Postconditions

The owner is able to see a dashboard that breaks down the amount of contribution being done to a resource by user.

Methodologies

- Assumptions:
  - One owner employs / manages multiple Maintainers / Operators
  - Each Maintainer / Operator only supports one Owner
  - Each Maintainer / Operator may support several Resources and support levels may not be evenly distributed because they may have been the contingency / fallback for some Resources
- Context: This panel's goal is to quickly tell the Owner who has been contributing / supporting how much and which Resource. There is a possibility that an Operator / Maintainer mainly focuses on 2 of 7 instruments they were requested to support. Looking at support levels for each Resource might be tedious. A single view might provide the Owner with a more wholesome picture.
- Table:
  - with columns:
    - Resource
    - Resource Classification
    - Operator / Maintainer
    - Campaign ID
    - Campaign date and time
    - Campaign classifier(s) - E.g. Materials, Biology, Manufacturing, etc.
    - Primary / Contingency Resource - whether this Resource was requested as the primary Resource for the Campaign or ended up being used as a fallback / contingency / secondary Resource
    - Campaign outcome - Success / Error / Aborted by User
  - Table should allow Owner to group a set of Resources if they own several different kinds of Resources such as scanning probe microscopes, electron microscopes, ... Typically disjoint set of Operators / Maintainers for each group of similar / identical Resources.
  - Owner should be able to filter by any of the columns
  - When at most three columns are remaining, INTERSECT should allow the Owner to plot column A vs Column B with column C as the legend. The Owner should be allowed to pick which columns in the table make up the axes and the curves / legend.
  - Plots should answer questions such as:
    - Operational:
      - Operator / Maintainer:
        1. How much is Operator X contributing per week for all Resources?
           (a) Line / bar graph with time on X axis and number of campaigns on the Y axis
        2. Who are the primary Operator(s) who support Resource Y?
           (a) Bar chart with Operator on the X axis and number of campaigns on the Y axis

(b) OR line graph with time on the X axis and number of Campaigns on the Y axis with each Operator represented as a different line

(c) OR stacked bar graph with time on X axis and number of Campaigns on Y axis with a stack of colored bars for each time step representing number of Campaigns supported by each Operator

3. Who are the primary Operator(s) who support Resource Classification P?
   (a) Family of lines
4. Is there a correlation between the Campaigns that are a success and the Operator / Maintainer?

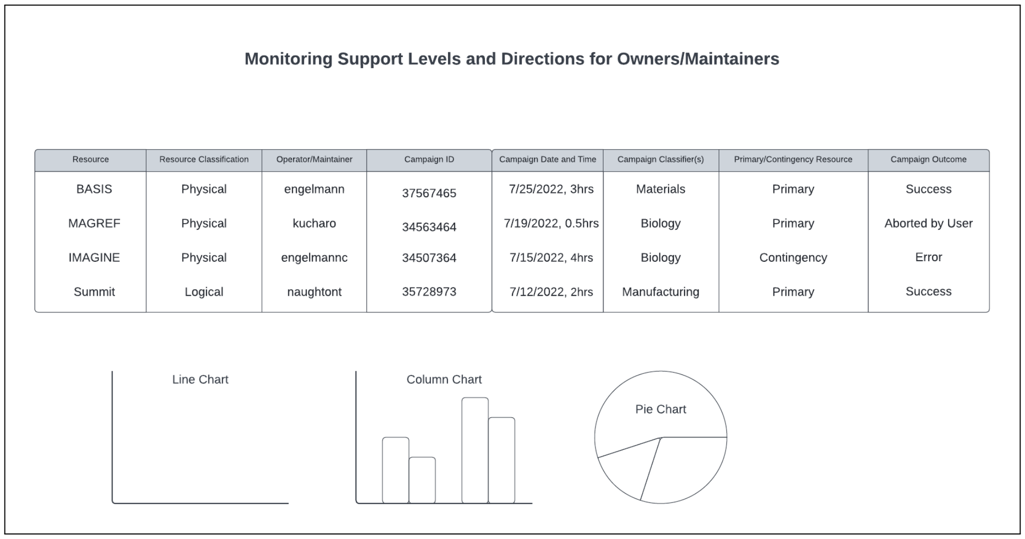
· Efficacy / Reliability of Resources:
1. Which Resources have a higher proportion of successful Campaigns as opposed to errors?

· Utilization rate
1. Number of Campaigns/< *time* >
2. Hours in use/< *time* >
3. < *time* > = day, week, month, year

· Utilization as primary vs. secondary Resource -
1. is this Resource primarily being used in place of another unreliable Resource or do Users request this Resource when composing their Campaign
2. Can also break down secondary into:
   (a) Declared as secondary Resource in Campaign
   (b) Owner modified Campaign(s) to use this alternate Resource instead of the Primary / secondary Resources declared in the Campaign
3. Stacked (composition) bar graph ($primary + secondary = 1.0$) with X-axis indicating time and Y-axis indicating percentage of utilization type
4. Family of lines - X-axis is time, Y-axis is percentage of primary usage. Different lines denote different Resources. Perhaps this sort of a graph illustrates to the Owner:
   (a) which Resources are picking up the slack when one or more are unavailable?
   (b) Which Resources are simply popular in Campaigns compared to others in the same Resource set?

∗ Scientific
   · What domains use this Resource the most? What have the trends been over time?
   1. Set of bar graphs - one per unit time. Each graph could show counts of Campaigns for each of the top N scientific domains.

An example interface is depicted in Figure A.27


Monitoring Support Levels and Directions for Owners/Maintainers

Resource	Resource Classification	Operator/Maintainer	Campaign ID	Campaign Date and Time	Campaign Classifier(s)	Primary/Contingency Resource	Campaign Outcome
BASIS	Physical	engelmann	37567465	7/25/2022, 3hrs	Materials	Primary	Success
MAGREF	Physical	kucharo	34563464	7/19/2022, 0.5hrs	Biology	Primary	Aborted by User
IMAGINE	Physical	engelmannc	34507364	7/15/2022, 4hrs	Biology	Contingency	Error
Summit	Logical	naughtont	35728973	7/12/2022, 2hrs	Manufacturing	Primary	Success

Figure A.27. Owners can see who is contributing to what resource.

Owner views a listing of all resources in INTERSECT that they manage.

Preconditions

The user is logged into INTERSECT with the owner role assigned and specific resources assigned to them.

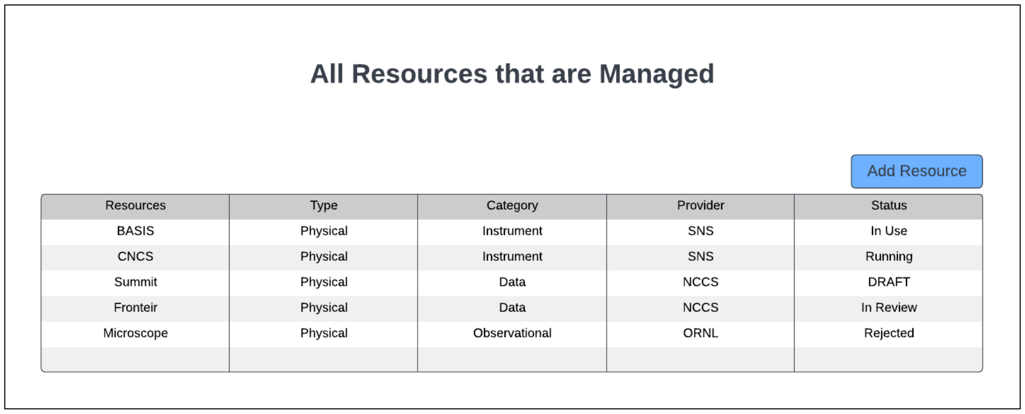
Postconditions

The owner is shown a list of resources that they manage with an ability to add resources and view their current status.

Methodologies

- This listing could be segregated by the type of the resource (e.g. observational resources, visualization application, data analysis application)
  - This view could have bare minimal information about the resource in the interest of space
  - Clicking on an existing resource allows management of said resource (see below)
- This listing would also list Resources that are in "draft" mode - something that the team is putting together but has not yet submitted to INTERSECT Administrator(s)
- This listing would also list Resources that have been submitted to the INTERSECT Administrator(s) and are currently being reviewed
- This listing would also list Resources that have been rejected by the INTERSECT Administrator after reviews. Perhaps this allows the Owner to copy paste content to newer applications, learn from failures, etc.
- Button to add new resource (see below)

An example interface is depicted in Figure A.28



Resources	Type	Category	Provider	Status
BASIS	Physical	Instrument	SNS	In Use
CNCS	Physical	Instrument	SNS	Running
Summit	Physical	Data	NCCS	DRAFT
Fronteir	Physical	Data	NCCS	In Review
Microscope	Physical	Observational	ORNL	Rejected

Figure A.28. A list of resources separated by Type, Category, and Provider.

Owner transfer ownership of a Resource to another User

Owner Edits configuration for a Resource

Preconditions

The user is logged into INTERSECT with the owner role assigned and currently viewing the detailed view of a resource seen in Figure A.12

Postconditions

The owner can see a list of the title and date/time of an experiment along with the requestor. The owner can also see who is supporting it and has the ability to temporarily or permanently remove the resource.

Methodologies

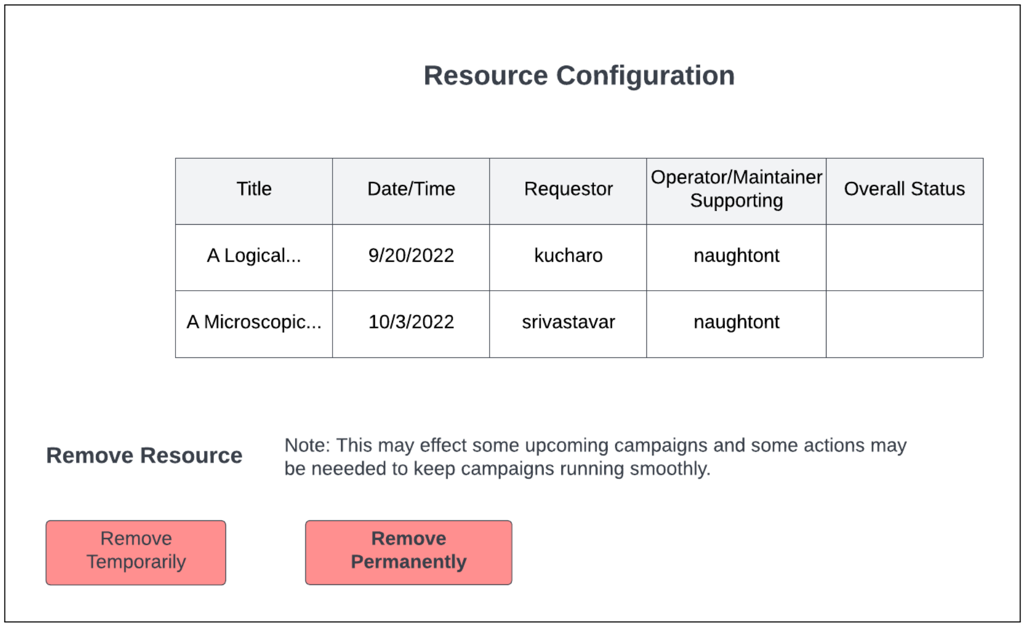
- edit configuration for an approved resource
-
- Owner manages a single resource from the detailed view about this Resource, already on INTERSECT
  - Owner views (succinct listing of) Campaigns - past, present, and scheduled that will use this Resource
    - Short title
    - Date and time
    - Requestor
    - Operator / Moderator who was / is / will be supporting this Campaign
    - Overall status color -
      - Scheduled Campaigns could respect the earlier defined options
      - Active Campaigns could be marked as “OK” or “green”
      - Past Campaigns -
        1. Cancelled / aborted by user - orange?
        2. Successful - Green
        3. Error - red
    - Clicking on one would point to the Detailed view of the Campaign
  - Owner removes resource from INTERSECT from the listing of managed resources
    - Temporarily
      - Resource is still visible in the catalog of resources but is marked as unavailable
        1. Perhaps the resource is moved lower in the search results / catalog listing with a greyed out title and thumbnail
    - Permanently
      - Resource is removed from INTERSECT in a permanent way
    - Regardless:
      - Dashboard asks the Owner if they are sure they want to do this
      - Dashboard asks Owner the reason for removing the Resource with common options such as “routine maintenance”, “major upgrades”, “Faulty / broken”, and “Other”.
      - INTERSECT adds this as an event to the Resource if the resource is
      - Dashboard will also notify the Owner that this is going to affect some upcoming Campaigns. Owner is asked to take actions to keep Campaigns moving smoothly. Owner can take a case-by-case approach or a bulk approach (same decision for all campaigns)

1. Owner can move the Campaigns to use an alternate resource if a reasonable substitute is available.
   (a) INTERSECT dashboard adds to the log of the specific campaign that the resource has been swapped.
   (b) User is notified via email
   (c) User can choose to accept the change, cancel the Campaign, or modify the Campaign configuration as they see fit.
2. If no substitutes are available:
   (a) INTERSECT marks the campaign with an error "unable to move forward"
   (b) User is notified via email

· INTERSECT emails all "regular" / past Users of this Resource, who do not have a scheduled Campaign using this Resource, that:
1. the Resource has been taken down
2. Alternate solution available / not available

- ~~Owner triages / assigns reassigns Maintainer / Operator for a given Campaign~~
  * ~~Assumption - INTERSECT will not automatically assign the Maintainer / Operator.~~
  * ~~Assumption - We may not have more than a few Campaigns per year and we may likely only have a single Operator / Moderator per Resource~~
  * ~~Question - Do we really want the Owner to have to deal with moderation / triaging? Alternatively, should INTERSECT ask the multiple Operators / Maintainers to decide amongst themselves as to who will support a given Campaign - How do we ensure equity / workload distribution among the~~
  * ~~Nuance - Variables: which Resource; which Campaign; which Operator~~
- Owner monitors this Resource using tables and/or graphs:
  * In most cases - Simple line / bar graph with counts (number of Campaigns) on the y-axis and time on the x-axis. Alternatively, simpler histogram / counts might be plenty. If neither are possible, tables would do.
  * Perhaps the same table that was presented outside the per-Resource view could be filtered to only show rows specific to this Resource. This is just a convenience for the Owner.
  * Operational
    · Up-time (time that the Resource was available for use)
    1. As a percentage per time period. E.g. - 96% in February 2022.
       (a) All duration that the Resource is marked as "Down" is counted
    2. Number of internal failures/<time> where failures
    · Are they meeting INTERSECT's SLA requirements? - this could be a checklist to make it easier on the Owner to track
  * Technical
    · Mean performance (multiple parameters) as relevant for this Resource (doesn't matter for visualization widgets for example)
    1. E.g.
    · Failure reason popularity
  * Scientific:
    · Capability / Mode popularity - simple bar graph
    · Campaign template popularity
  * Others?

An example interface is depicted in Figure A.29



Resource Configuration

Title	Date/Time	Requestor	Operator/Maintainer Supporting	Overall Status
A Logical...	9/20/2022	kucharo	naughtont	
A Microscopic...	10/3/2022	srivastavar	naughtont	

Remove Resource

Note: This may effect some upcoming campaigns and some actions may be neeeded to keep campaigns running smoothly.

Figure A.29. Options are given to temporarily or permanently remove the resource.

A.3 OPERATOR / MAINTAINER (OM)

OM views notifications in the Notification Panel

Preconditions

The user is logged into INTERSECT with the operator role assigned.

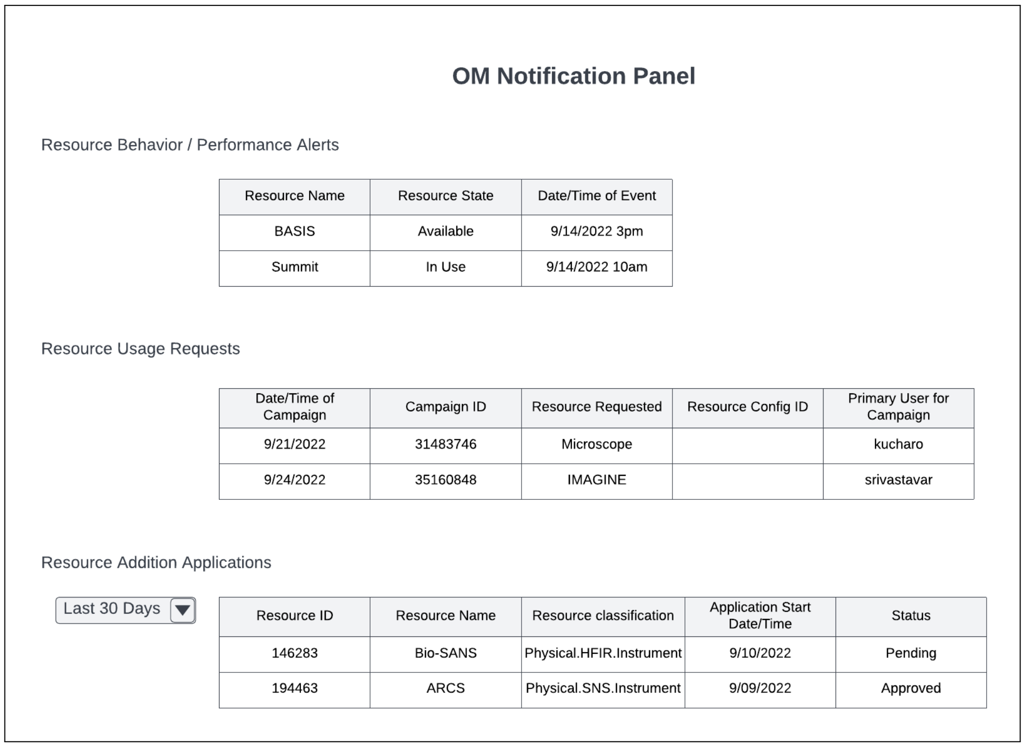
Postconditions

The operator can see a panel with resource behavior, resource usage requests, and resources addition applications for a resource(s) they manage.

Methodologies

- Notification Panel would be divided into at least 3 segments:
  - Other (relevant to all Users)
  - Resource behavior / performance alerts - whenever Resource is not in good health or active usage
    - The listing table could have the following columns:
      - Resource Name
      - Resource state - identifier,
      - Date and time of event
  - Resource usage requests (for upcoming Campaigns) -
    - The listing table could have the following columns:
      - Date and time of Campaign
      - Campaign ID
      - Resource requested
      - Resource configuration identifiers if any
      - Primary User for Campaign
    - Additional information regarding the request can be obtained from the detailed view of the Campaign
    - Notification will be dismissed only when the Resource is no longer blocked by this OM or whenever the OM manually dismisses the notification
  - Resource addition applications - Resources that an Owner would like to add to INTERSECT and has listed this Operator / Moderator as a
    - This listing could have the following columns:
      - Resource ID
      - Resource Name
      - Resource Classification
      - Application start date and time

An example interface is depicted in Figure A.30


OM Notification Panel

Resource Behavior / Performance Alerts

Resource Name	Resource State	Date/Time of Event
BASIS	Available	9/14/2022 3pm
Summit	In Use	9/14/2022 10am

Resource Usage Requests

Date/Time of Campaign	Campaign ID	Resource Requested	Resource Config ID	Primary User for Campaign
9/21/2022	31483746	Microscope		kucharo
9/24/2022	35160848	IMAGINE		srivastavar

Resource Addition Applications

Resource ID	Resource Name	Resource classification	Application Start Date/Time	Status
146283	Bio-SANS	Physical.HFIR.Instrument	9/10/2022	Pending
194463	ARCS	Physical.SNS.Instrument	9/09/2022	Approved

Figure A.30. Operator/Maintainer notification panel.

OM works with other OMs and Owner to fill out and submit an application to add a Resource to INTERSECT

OM Monitors Resource(s) they maintain on a routine basis

Preconditions

The user is logged into INTERSECT with the operator role assigned.

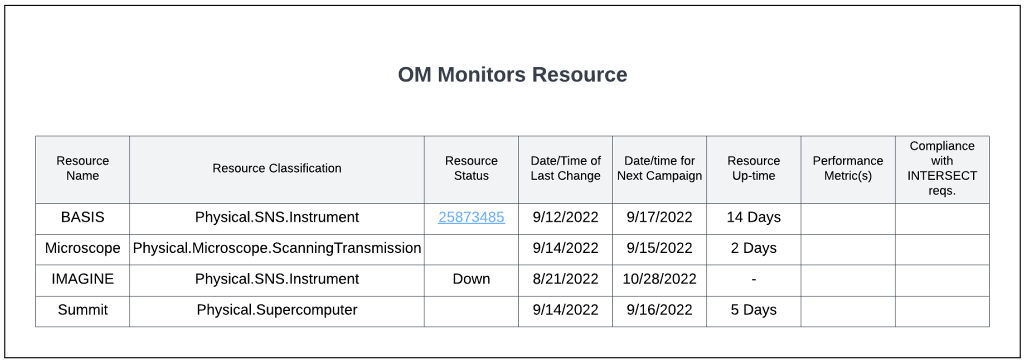
Postconditions

The operator will see a table with resource classification, status, and dates being used for campaigns along with other resource health stats.

Methodologies

- This view should provide a sortable table
  - The table should have the following columns:
    - Resource Name
    - Resource Classification - e.g. "Physical.Microscope.ScanningTransmissionElectronMicroscope", "Physical.3DPrinter.Blah", etc.
    - Resource state - available / in-use / degraded / down
    - Resource state identifier or notes
      - Nothing in the case of "available"
      - Campaign ID in case of "in-use" with link to Campaign details
      - In case of "down" or "degraded' - few-word summary of the log (instead of a long description. More details could be obtained from the detailed view of the Resource's logs / history.
      - Detailed log messages would only be visible to the Owner and OMs of the Resource and not to the User). OM should be allowed to edit this identifier message
    - Date and time of last change in state
    - Date and time for next Campaign
    - Up-time
    - Performance metric(s)
    - Compliance with INTERSECT requirements - this requires more thought.
- This view should also provide graphical representations of the state / history of individual resources?

An example interface is depicted in Figure A.31



OM Monitors Resource

Resource Name	Resource Classification	Resource Status	Date/Time of Last Change	Date/time for Next Campaign	Resource Up-time	Performance Metric(s)	Compliance with INTERSECT reqs.
BASIS	Physical.SNS.Instrument	25873485	9/12/2022	9/17/2022	14 Days		
Microscope	Physical.Microscope.ScanningTransmission		9/14/2022	9/15/2022	2 Days		
IMAGINE	Physical.SNS.Instrument	Down	8/21/2022	10/28/2022	-		
Summit	Physical.Supercomputer		9/14/2022	9/16/2022	5 Days		

Figure A.31. Operator/Maintainer monitors resource.

OM updates Resource information / configuration via the Detailed View for a given Resource

Preconditions

The user is logged into INTERSECT with the operator role assigned.

Postconditions

The status of the resources has been updated or the updates have been sent to the owner for approval.

Methodologies

- Some changes require approval by the Owner of the Resource
- All changes will be notified to the Owner
- OM may update the current status of the Resource, simplify lengthy log messages to a single error category that may help other OMs or the Owner quickly understand the issue with the Resource, OM may also update API calls or addresses / ports for Resource controller and provisioning services.

OM takes down a Resource via the Detailed View

Preconditions

The user is logged into INTERSECT with the operator role assigned. They are currently looking at the detailed view of a resource seen in Figure A.12. The operator's version of the detailed view of a resource will have a "decommission" and "suspend" button up in the top right corner.

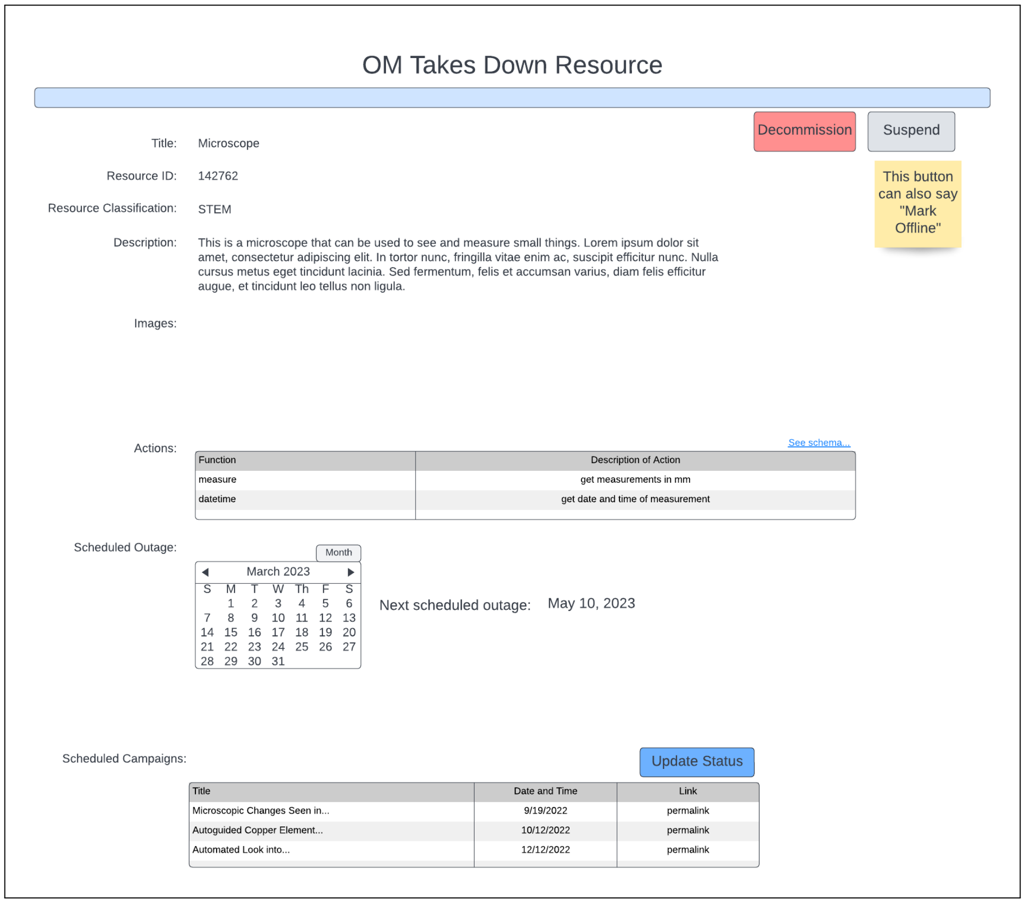
Postconditions

The operator will have to either move campaigns to use a similar resource or would have to send an error on campaigns currently using that resource. This would require sending a notification to schedulers.

Methodologies

- OM would click on a "Suspend" or "Mark offline" button in the Detailed View to temporarily make the Resource unavailable for INTERSECT Campaigns.
- Should the downtime affect any upcoming Campaigns, the OM either takes a bulk decision or a case-by-case decision on the Campaigns, similar to the Owner:
  - OM moves a Campaign(s) to use a similar / equivalent Resource that they also maintain
    - Given that the OM has edit capabilities to the step in the workflow configuration pertaining to the Resource(s) that the OM controls, the OM can swap the affected Resource with an alternate Resource. Thereby, the Campaign would not be using a Resource that is unavailable.
    - This swap is noted in the logs for the Campaign and the OM notes the reason for the swap of Resource in the logs for full transparency.
  - OM throws an error to the Campaign letting the User know that they need to either schedule the Campaign for a later time or consider using a different Resource, outside the control of the OM.
    - The Detailed View of the Campaign allows OMs of Resources (expected to be used in the Campaign) to mark a Resource request as unfulfillable.

An example interface is depicted in Figure A.32



OM Takes Down Resource

Title: Microscope

Resource ID: 142762

Resource Classification: STEM

Description: This is a microscope that can be used to see and measure small things. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In tortor nunc, fringilla vitae enim ac, suscipit efficitur nunc. Nulla cursus metus eget tincidunt lacinia. Sed fermentum, felis et accumsan varius, diam felis efficitur augue, et tincidunt leo tellus non ligula.

Images:

Actions:

Function	Description of Action
measure	get measurements in mm
datetime	get date and time of measurement

Scheduled Outage:

March 2023

S	M	T	W	Th	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Next scheduled outage: May 10, 2023

Scheduled Campaigns:

Title	Date and Time	Link
Microscopic Changes Seen in...	9/19/2022	permalink
Autoguided Copper Element...	10/12/2022	permalink
Automated Look into...	12/12/2022	permalink

Figure A.32. This take down takes places in the detailed view.

OM handles requests to set-up Resource(s) for Campaigns / OM Handles Request for Project to Gain Access to Resource - these are two different scenarios

Preconditions

The user is logged into INTERSECT with the operator role assigned and looking at the detailed view of a Campaign seen in Figure A.7.

Postconditions

The operator will look at the resource within the campaign and update the status of the campaign as successful, infeasible, or failure, with the ability to type any alternates.

Methodologies

- OM views and sorts the listing of requests either from the notification panel or from the detailed view for each Resource
- OM reads details about the Campaign from the detailed view of the Campaign (accessible from the list of upcoming Campaigns using the Resource, in the Resource's Detailed View), including specifics for the Resource(s) that need to be set up.
- At this stage, OM takes one of the following actions:
  - OM communicates inability or infeasibility to setup Resource(s) by responding to the Campaign usage request and rejecting the Campaign
  - OM communicates alternate setup configurationS
  - OM sets up Resource according to request
- OM updates the Campaign status by marking setup as successful / infeasible / failure / suggests alternate upon setting up the Resource(s) offline
-
- The interface for fulfilling the requests for different Resources for a given Campaign could take significant inspiration from that of a "Pull Request" in GitHub or GitLab that require multiple individuals to review / approve the Pull Request.
  - The User's submission of a Campaign would be equivalent to a Pull Request
  - OMs would communicate about progress in setting up / caveats regarding the use of / inability to set up their Resource(s) using notes / comments, much like the comments on GitHub for a Pull Request.
  - OMs or Users could cancel the Campaign similar to closing a Pull Request using buttons at the very bottom
  - Similar to the concept of CI / CD pipeline checks, the Campaign could have multiple Resource(s) and other checks with independent statuses (ready, in preparation, etc.) and corresponding colors. The Campaign would also have an aggregate readiness rating that would be visible from a higher level listing view.
  - Perhaps to keep the Campaign Detailed View less cluttered, we could break down all the information into tabs or panes (General / high-level, Configuration, Status, Real-Time Dashboard) that would not individually overwhelm the User or OM.
- Operator synchronizes their calendar with INTERSECT?
  - Operator specifies the times that they are nominally available manually.
  - How would they specify this and where?
  - What happens to Operators outside ORNL?

An example interface is depicted in Figure A.33

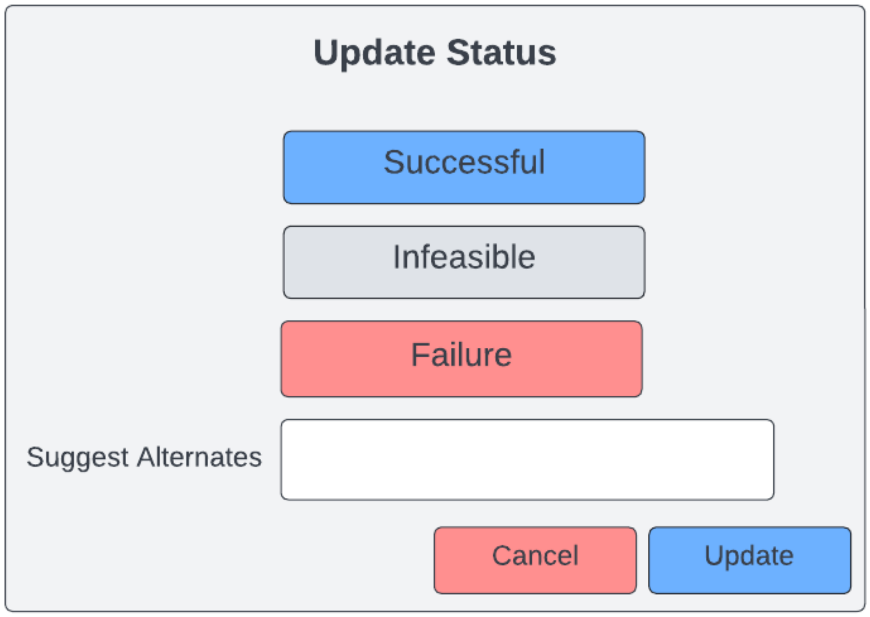


Figure A.33. Alternates can be suggested if the three choice do not suffice.

A.4 ADMINISTRATOR

Monitors all Resources

- Administrator monitors all Resources in addition to viewing requests to add Resources to INTERSECT:
  - Resource is assigned an ID when Owner composes form to create new Resource
  - Table with the following columns:
    - Resource ID - link that takes the Administrator to the Detailed View about the Resource
    - Resource Name - e.g. “SNS DGX 2” / Title (Abbreviated)
    - Resource Classification - Physical.Microscope.ScanningTransmissionElectronMicroscope or Logical.Visualization.2DImage
    - Owner
    - Organization
    - Location
    - Date added / Requested (to be added to INTERSECT)
    - Status: Active, Down, In use, Requested.
    - Compliance: Good, Moderate (e.g. - performance in one or more metrics may have dropped below ideal / promised levels but still functional), Failing (failing in at least one of the non-negotiable metrics described in the SLA)
  - The Listing View should allow the Administrator to easily filter by most columns like the Resource Classification, Owner, Location, Date, Status etc.
    - The Resource Classification is meant to provide increasing levels of granularity within the same topic without adding more columns to the table.
  - The Listing View should allow the Administrator to easily sort by any of the columns in the table
  - This View should also allow the Administrator to do basic visual analytics on the contents of the (filtered) table to answer questions such as:
    - Number of Resources added every month or year
    - Pie chart or similar distribution illustrating number of Resources by Organizations that own them
    - Distribution of Resources by Resource Classification (maybe just by the first level that varies within the current listing).
    - Distribution of the Compliance or Status of Resources by Organization / Owner or Classification.
- Administrator adds Resource to INTERSECT ecosystem:
  - Upon clicking on a link, the Administrator is taken to the Detailed View for the Resource.
  - The Detailed View of the Resource shows all relevant details regarding the Resource, including:
    - Background and other top-level details
    - Configuration information (not visible to Users, only visible to Operator / Maintainer(s), Owner, and Administrators)
    - a GitHub style view of communication between the Owner, Operator(s) / Maintainer(s) and Administrator(s). The Administrator can use this tool to communicate edits to the configuration, ask questions, make comments, etc.
      - These conversations are not visible to end Users of the Resource

            - Upon acceptance of this request, the first public log is automatically created that illustrates when the Resource was officially added to INTERSECT.
        - CI/CD like checklist that illustrates which of INTERSECT's requirements have been complied
        - Buttons:
            - to run the compliance checklist
            - Reject request
            - Approve request
- Administrator monitors all Users in INTERSECT in addition to managing permissions / roles using the User Management Panel:
    - Table with following fields:
        - Username (links to User profile).
        - Full name (Also links to same User profile)
        - Organization (applies filter on this specific Organization)
        - Role - User / Operator / Maintainer / Administrator
        - Requested role (populate only when different from existing role and only when User has requested a role change).
        - Role start date (could serve as the time stamp for the last time this person's role had changed)
        - Role end date
        - Button - "change role"
        - Date Added - useful for tracking growth rate
        - Last active - last date that this User used INTERSECT
        - Active days - number of days this User used INTERSECT. This might be useful to track user retention and desire to continue to use INTERSECT
    - The Listing View should allow the Administrator to easily filter by current role, requested roles (selects only those users who have requested role changes), organization, etc.
    - The Listing View should allow the Administrator to easily sort by any of the columns in the table
    - This View should also allow simple visual analytics such as:
        - Number of new Users added to INTERSECT per month
        - Number of active Users per month
        - Distribution of Users by Organization
        - Distribution of Users by Role
- Administrator monitors state of backbone INTERSECT microservices and central infrastructure

B INTERSECT MESSAGE SCHEMA

Listing 1. INTERSECT XML SCHEMA.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2021 rel. 2 (x64) (http://www.altova.com) by Olga Kuchar (Oak Ridge
     National Lab) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:vc="http://www.w3.org/2007/XMLSchema-
     versioning" elementFormDefault="qualified" attributeFormDefault="unqualified" vc:minVersion="
     1.1">
        <xs:element name="ScienceEcosystemMessages">
                <xs:annotation>
                        <xs:documentation>Top element for the INTERSECT System-of-Systems Messages.</
                             xs:documentation>
                </xs:annotation>
                <xs:complexType>
                        <xs:choice>
                                <xs:element name="CommandMessage" type="CommandMessageType">
                                        <xs:annotation>
                                                <xs:documentation>Command message type. Requires action
                                                     for receiver. Must be responded to by an
                                                     acknowledgement. Immediate response expected with
                                                     deferred processing.</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="AcknowledgeMessage" type="AcknowledgeMessageType">
                                        <xs:annotation>
                                                <xs:documentation>Acknowledgement message type. Response
                                                     to CommandMessage. Response is Accepted/Rejected.</
                                                     xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="RequestMessage" type="RequestMessageType">
                                        <xs:annotation>
                                                <xs:documentation>Request message type. Inquisitive for
                                                     receiver. Must be responded to by a reply. Response
                                                     expected after processing is complete. </xs:
                                                     documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="ReplyMessage" type="ReplyMessageType">
                                        <xs:annotation>
                                                <xs:documentation>Reply message type. Response to
                                                     RequestMessage. Response is detailed.</xs:
                                                     documentation>
                                        </xs:annotation>
                                </xs:element>
                                <xs:element name="EventMessage" type="EventMessageType">
                                        <xs:annotation>
                                                <xs:documentation>Notification message type. Asynchronous
                                                      and does not need a response or acknowledgement.
                                                     Examples include heartbeat message, log message,
                                                     status message.</xs:documentation>
                                        </xs:annotation>
                                </xs:element>
                        </xs:choice>
                </xs:complexType>
        </xs:element>
        <xs:element name="SourceID">
                <xs:annotation>
                        <xs:documentation>The sender ID of this message. This is the ID of the
                             component or system at which this message was formulated.</xs:
                             documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="TargetID">
        <xs:annotation>
            <xs:documentation>The receiver ID of this message. This is the ID of the component or system at which this message was sent.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="MessageID" type="xs:ID">
        <xs:annotation>
            <xs:documentation>The ID of this message. If "Command" or "Request", this is the ID used in the "Ack" or "Reply" message.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="MessageCreationDateTimeUTC" type="xs:dateTime">
        <xs:annotation>
            <xs:documentation>The UTC date and time that this message instance was created.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="CommandMessageType">
        <xs:annotation>
            <xs:documentation>Base type for a CommandMessage.</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="MessageID"/>
            <xs:element ref="MessageCreationDateTimeUTC">
                <xs:annotation>
                    <xs:documentation>The UTC date and time that this message instance was created.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element ref="SourceID">
                <xs:annotation>
                    <xs:documentation>The sender ID of this message. This is the ID of the component or system at which this message was formulated.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element ref="TargetID">
                <xs:annotation>
                    <xs:documentation>The receiver ID of this message. This is the ID of the component or system at which this message was sent.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element ref="MessageSchemaURI"/>
            <xs:element ref="MessageType" minOccurs="0"/>
            <xs:element ref="MessageName" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>The name for the given message type.</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element ref="MessageVersion" minOccurs="0"/>
            <xs:element ref="MessageContent">
                <xs:annotation>
                    <xs:documentation>The actual message content, deciphered by either MessageSchemaURI or (MessageType, MessageName, MessageVersion).</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="MessageSchemaURI" type="xs:anyURI">
```

```
            <xs:annotation>
                <xs:documentation>The URL of the schema to decipher the MessageContent element
                    .</xs:documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="MessageType" type="xs:string">
            <xs:annotation>
                <xs:documentation>The type of message - enumerated list. See
                    ScienceEcosystemMessageTypes.xsd for the current listing.</xs:
                    documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="MessageName">
            <xs:annotation>
                <xs:documentation>The name for the given message type.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:minLength value="1"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="MessageVersion">
            <xs:annotation>
                <xs:documentation>The version of the message type being used in the
                    MessageContent element.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:decimal">
                    <xs:minInclusive value="0"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:element name="MessageContent" type="xs:base64Binary">
            <xs:annotation>
                <xs:documentation>The actual message content, deciphered by either
                    MessageSchemaURI or (MessageType, MessageName, MessageVersion).</xs:
                    documentation>
            </xs:annotation>
        </xs:element>
        <xs:element name="ResponseToMessageID" type="xs:ID">
            <xs:annotation>
                <xs:documentation>The MessageID relating to the original request.</xs:
                    documentation>
            </xs:annotation>
        </xs:element>
        <xs:complexType name="AcknowledgeMessageType">
            <xs:sequence>
                <xs:element ref="MessageID"/>
                <xs:element ref="MessageCreationDateTimeUTC"/>
                <xs:element ref="SourceID"/>
                <xs:element ref="TargetID"/>
                <xs:element ref="ResponseToMessageID"/>
                <xs:element ref="ResponseSummary"/>
                <xs:element ref="MessageSchemaURI"/>
                <xs:element ref="MessageType" minOccurs="0"/>
                <xs:element ref="MessageName" minOccurs="0"/>
                <xs:element ref="MessageVersion" minOccurs="0"/>
                <xs:element ref="MessageContent"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="ReplyMessageType">
            <xs:annotation>
                <xs:documentation>Base type for a ReplyMessage.</xs:documentation>
            </xs:annotation>
            <xs:sequence>
                <xs:element ref="MessageID"/>
                <xs:element ref="MessageCreationDateTimeUTC"/>
                <xs:element ref="SourceID"/>
                <xs:element ref="TargetID"/>
                <xs:element ref="ResponseToMessageID"/>
                <xs:element ref="ResponseSummary"/>
                <xs:element ref="MessageSchemaURI"/>
                <xs:element ref="MessageType" minOccurs="0"/>
                <xs:element ref="MessageName" minOccurs="0"/>
                <xs:element ref="MessageVersion" minOccurs="0"/>
                <xs:element ref="MessageContent"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="ResponseSummary">
            <xs:annotation>
                <xs:documentation>Enumerated type of SUCCESS or FAILURE only. More information
                    is provided in MessageContent element.</xs:documentation>
            </xs:annotation>
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="SUCCESS"/>
                    <xs:enumeration value="FAILURE"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
        <xs:complexType name="RequestMessageType">
            <xs:annotation>
                <xs:documentation>Base type for a RequestMessage.</xs:documentation>
            </xs:annotation>
            <xs:sequence>
                <xs:element ref="MessageID"/>
                <xs:element ref="MessageCreationDateTimeUTC"/>
                <xs:element ref="SourceID"/>
                <xs:element ref="TargetID"/>
                <xs:element ref="MessageSchemaURI"/>
                <xs:element ref="MessageType" minOccurs="0"/>
                <xs:element ref="MessageName" minOccurs="0"/>
                <xs:element ref="MessageVersion" minOccurs="0"/>
                <xs:element ref="MessageContent"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="EventMessageType">
            <xs:annotation>
                <xs:documentation>Base type for a EventMessage.</xs:documentation>
            </xs:annotation>
            <xs:sequence>
                <xs:element ref="MessageID"/>
                <xs:element ref="MessageCreationDateTimeUTC"/>
                <xs:element ref="SourceID"/>
                <xs:element ref="TargetID"/>
                <xs:element ref="MessageSchemaURI"/>
                <xs:element ref="MessageType" minOccurs="0"/>
                <xs:element ref="MessageName" minOccurs="0"/>
                <xs:element ref="MessageVersion" minOccurs="0"/>
                <xs:element ref="MessageContent"/>
            </xs:sequence>
        </xs:complexType>
    </xs:schema>
```