

SCP-Lab: A Protocol for Laboratory Capability Access and Coordination

Physical Intelligence Team, Shanghai AI Laboratory

Scientific agents support hypothesis generation and experimental planning, but validating their proposals in the natural sciences often requires physical experiments. Autonomous laboratories aim to close the loop between planning, execution, and experimental feedback, yet heterogeneous equipment and interfaces complicate capability integration and reuse. We propose SCP-Lab as a laboratory-facing extension of the Science Context Protocol. It organizes laboratory access through five levels of exposed control granularity, from established workflows to actuator actions. These levels clarify the choices available to a user and the responsibilities retained by a facility. A common interface supports capability discovery, task submission, and execution feedback, with records that link physical completion evidence to the relevant tasks, samples, and results. Laboratory cases spanning chemistry, materials science, and the life sciences illustrate workflow and device access, embodied operations, and coordination across laboratories. The design aims to reduce integration and expansion costs, broaden experimental scope, and support collaboration across facilities and regions.

Project repository: <https://github.com/InternRobotics/SCP-Lab>

Keywords: Scientific agents; Autonomous laboratories; Laboratory access protocols; Laboratory interoperability; Control granularity

1. Introduction

Scientific discovery advances through a continuing exchange between ideas and evidence. Specialized AI models are accelerating this exchange by expanding the scale of scientific prediction, design, and search [1, 2, 3, 4]. Scientific agents extend these advances into research workflows that connect knowledge retrieval, hypothesis generation, and experimental planning with feedback from scientific tools [5, 6, 7]. Such workflows have demonstrated progress in AI research and mathematics, where computational experiments and automated evaluators provide direct feedback on candidate solutions [8, 9, 10]. In the experimental natural sciences, however, computational proposals must ultimately be tested against physical evidence, often through wet-lab validation.

Reliance on manual execution can make this validation difficult to reproduce and scale: procedures depend on operators' experience, and manual experimentation is time-consuming [11]. Autonomous laboratories aim to address this bottleneck by continuously translating high-level goals into executable plans and trustworthy experimental evidence [12, 13, 14, 15]. Yet experimental equipment varies widely across scientific fields and laboratories [16, 17]. Some platforms run complete experimental workflows and allow users to adjust only specified parameters, while other equipment supports selecting individual operations and changing their sequence. This heterogeneity makes it difficult to reuse existing experimental capabilities or coordinate equipment across laboratories without repeated integration work [18, 19].

Shared interfaces offer a way to connect heterogeneous resources. A general example is the Model Context Protocol (MCP), which standardizes tool and data access for AI applications [20]. The Science

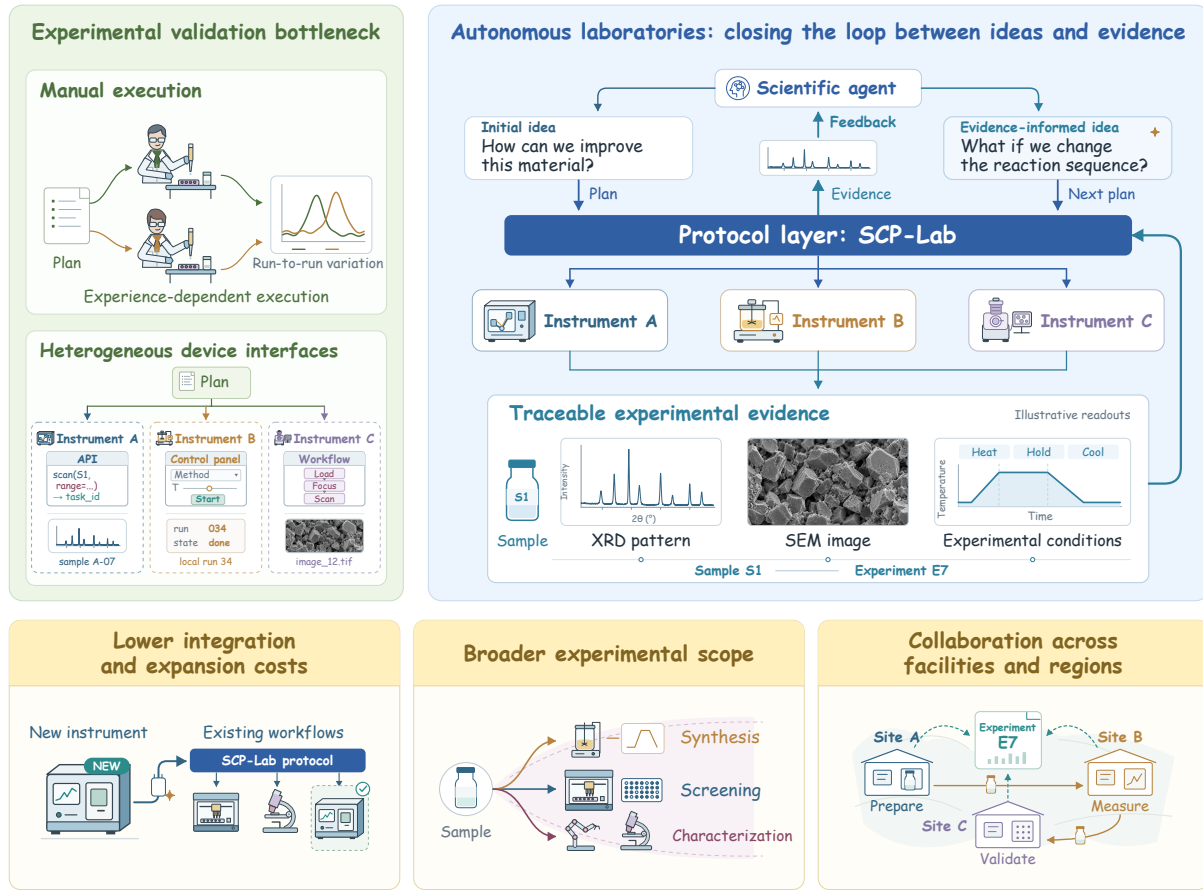


Figure 1 | **Connecting scientific ideas and experimental evidence through SCP-Lab.** Manual experimentation is time-consuming and depends on operator experience, while heterogeneous equipment complicates integration and reuse. SCP-Lab connects scientific agents with laboratory capabilities to support a closed loop in which experimental evidence informs new ideas and subsequent experiments. The design aims to reduce integration costs, broaden experimental scope, and support collaboration across laboratories.

Context Protocol (SCP) applies this approach to scientific resources, including software, models, datasets, and physical instruments [21]. The Model Hardware Standard (MHS) research preview introduces standardized drivers and device discovery to support agent control and coordination across physical devices [22]. Together with laboratory interoperability standards [23, 24], these efforts provide a foundation for shared laboratory access. A common interface needs to accommodate different degrees of equipment control: users should know which experimental capabilities are available, what they can adjust, and how to obtain execution feedback and results.

To address these requirements, we propose SCP-Lab as a laboratory-facing extension of SCP. The protocol supports a closed loop between idea generation and experimental validation, in which experimental evidence helps refine hypotheses and guide subsequent experiments. The multilevel access framework clarifies which experimental operations users can invoke and which parameters or procedural steps they can adjust. Users can therefore reuse established workflows or customize procedures for new experiments within the control options their equipment provides. These capabilities are accessed through a common interface, while device-specific commands and physical execution are handled locally. This separation supports the reuse of verified adapters and workflows across

scientific agents, as well as new combinations of equipment functions and embodied operations. As summarized in Figure 1, the design aims to reduce integration and expansion costs, broaden experimental scope, and support collaboration across facilities and regions. Cases spanning chemistry, materials science, and the life sciences ground these design choices in existing experimental practice.

2. Related Work

Scientific Agents and Autonomous Experimentation. Work on autonomous experimentation connects scientific reasoning with physical testing [12, 25]. This research spans learning-guided reactivity searches and robotic synthesis [26, 27], as well as autonomous platforms for flow chemistry, protein engineering, and materials discovery [28, 29, 30, 15, 31, 32]. Agents based on large language models (LLMs) contribute natural-language reasoning and tool use to this effort, supporting chemical research [5, 33, 6] and hypothesis development in AI co-scientist [7] and Virtual Lab [34]. Natural-language interaction also supports experimental execution: ORGANA [35] and AILA [36] connect scientists’ requests to robotic chemistry tasks and atomic force microscopy, respectively. Across these settings, the link between scientific goals and physical experiments depends on the capabilities available through laboratory interfaces.

Laboratory Software and Workflow Orchestration. Laboratory control and orchestration software provides the execution support for these systems. The Chemputer [37] and Chemical Description Language (XDL) [38] translate procedure descriptions into equipment operations. ChemOS [39, 17] separates experiment planning from hardware-specific execution, while HELAO-async [40] and IvoryOS [18] coordinate workflows and instruments. Distributed studies extend coordination across facilities while retaining local execution [41, 42, 19], and UniLabOS [43] models laboratory actions and resources for this setting. SCP-Lab follows this modular approach, defining a common interface for the workflows and device functions exposed by existing laboratory software.

Protocols and Interoperability Standards. MCP [20, 44] provides general tool and data access, including experimental support for durable tasks in its November 2025 specification, while SCP [21] unifies scientific resource description and invocation. The MHS research preview [22] introduces standardized device drivers and discovery, with access through MCP, command-line interfaces, and code. PUDA [45] similarly combines command-line hardware access with structured experimental records. Laboratory standards provide more specific execution models: Standardization in Lab Automation (SiLA) 2 [23] defines observable commands and execution feedback, while the Laboratory and Analytical Device Standard (LADS) [24] covers device programs, execution state, and sample and result relationships. SCP-Lab focuses on the experimental control exposed through an interface. Its five-level model relates exposed control choices to execution responsibilities, physical completion evidence, and the sample context needed to interpret experimental results.

3. Architecture and Experimental Workflow

Figure 2 summarizes the roles of a scientific agent, a laboratory operating system, and the local execution layer at each laboratory. A scientific agent might propose the high-level plan: “*I want to synthesize zinc oxide nanoparticles using a hydrothermal method and investigate how reaction temperature affects their size.*” Executing this plan requires choices about the reactor, precursor quantities, reaction temperature and duration, and equipment for sample preparation and size characterization. The laboratory operating system selects suitable equipment, specifies these settings, and coordinates the resulting tasks through SCP-Lab. The local execution layer operates the equipment. As observations return, the scientific agent can evaluate the proposed relationship and decide which experiments to pursue next.

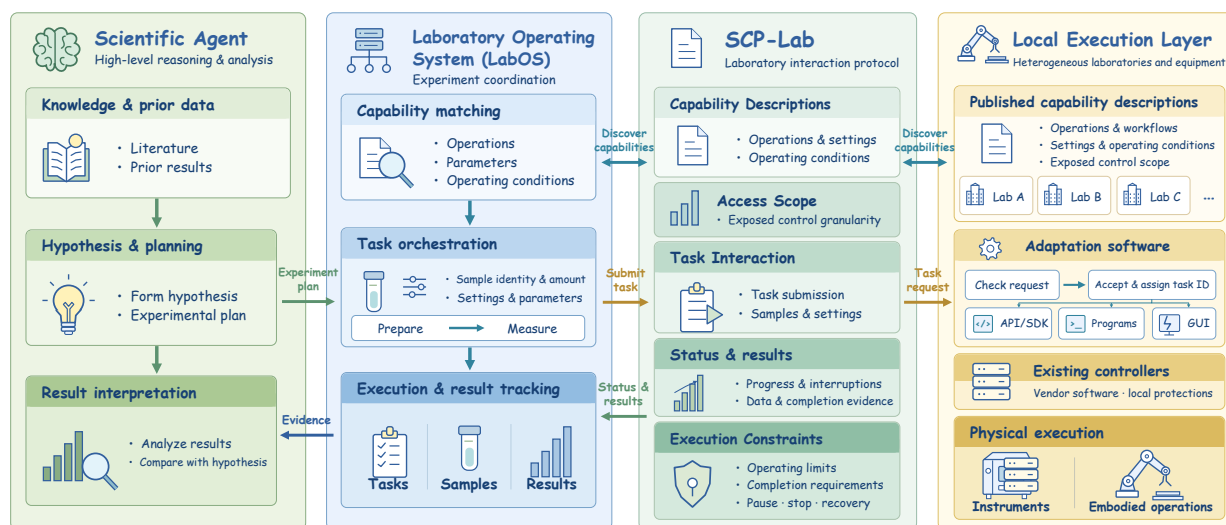


Figure 2 | **SCP-Lab in the experimental workflow.** A scientific agent develops high-level experimental plans and interprets returned evidence. A laboratory operating system selects capabilities and coordinates tasks through SCP-Lab, while local adaptation software and existing controllers execute the work. Capability discovery, task submission, and status and result feedback connect these roles across laboratories and support both equipment functions and embodied operations.

The workflow involves three forms of interaction. First, *discovering available operations* establishes what a laboratory can do and helps the operating system assign tasks to suitable resources while considering expected execution efficiency. Each laboratory describes the operations or workflows it offers; we call each of these a *capability*. The description states the available settings and operating conditions. The operating system can then select, for example, a reactor with a suitable temperature range and vessel capacity. Second, *submitting experimental tasks* specifies the work to be carried out. The operating system supplies sample information and settings for the selected operation, such as the reaction temperature and duration in the synthesis example. The local execution layer checks the request, assigns an identifier to accepted work, and starts execution when local conditions permit. Third, *receiving status and results* lets the operating system follow execution and obtain experimental data. The local execution layer reports progress and interruptions throughout the task and returns results as they become available. These records associate observations, such as temperature profiles and characterization data, with the relevant tasks and samples. The operating system uses progress reports to coordinate subsequent steps, while the scientific agent assesses the findings.

The *local execution layer* comprises adaptation software and the equipment's existing controllers. Its implementation depends on the manufacturer: some instruments expose an application programming interface (API) or a software development kit (SDK), while others can only be operated through the graphical user interface (GUI) of their control software. The adaptation software converts task parameters into the appropriate calls, imported programs, or GUI actions; the controllers carry out the physical work. The software also reads equipment status and data and associates them with the submitted task. We can reuse the same adaptation across scientific agents and experimental workflows because the local execution layer handles the manufacturer-specific interfaces.

Small-batch, exploratory experiments may not warrant a dedicated automated setup, and specialized equipment may be poorly suited to open-ended procedures. Embodied systems could support such work through flexible manipulation and the use of existing laboratory tools. Robotic chemistry already includes manipulating laboratory vessels and operating conventional instruments [13], as well

Table 1 | Five access levels for heterogeneous laboratory equipment. The levels describe the scope of the caller’s control; every level may expose adjustable parameters.

Level	Form of access	Scope of the caller’s choices
L1	Established workflow	Select a workflow and set its exposed experimental inputs and conditions; the internal steps and order remain fixed.
L2	Configurable workflow	Select and organize the steps or modules made available by the facility, which executes the submitted workflow.
L3	Single-device function	Invoke a function and adjust its exposed parameters while the local system retains the predefined internal steps and order.
L4	Internal device operations	Select, reorder, or repeat exposed steps within a device function, and set their parameters through supported commands or program configuration.
L5	Actuator action	Specify exposed motion or actuation commands, such as a pose, rail position, or gripper action.

as electrode polishing and procedures guided by visual feedback [35]. We therefore treat embodied operations as part of the experimental work that a laboratory can offer through SCP-Lab. For example, a laboratory could expose robotic electrode preparation alongside electrochemical measurement. The operating system would identify the electrode and select a supported preparation procedure; the robot would carry it out and report whether it completed or stopped early. The coordinator could request the measurement after preparation was confirmed and link both tasks to the same experiment. Sample transport and instrument loading can participate in this workflow alongside these experimental operations.

Coordination across laboratories must connect independently maintained resources. Each site publishes its own capability descriptions and access conditions, while the laboratory operating system selects operations and assigns tasks across sites. The operating system links each site’s task records to the scientific plan and collects results together with sample identities and experimental conditions. A study can thus draw on complementary resources across institutions and retain the origin and context of each contribution. Material transfer between sites adds transport and receipt to the workflow (Section 5). A caller may delegate a complete workflow or a finer experimental action, both within a laboratory and across sites. Section 4 defines the five levels of control granularity.

4. Five Levels of Control Granularity

Laboratory equipment differs in both its experimental functions and the control it makes available to users. An integrated station may accept only a complete workflow, while another system allows users to arrange experimental steps or alter a device’s internal procedure. We introduce five access levels to accommodate these different forms of access within SCP-Lab (Figure 3; Table 1). The levels span whole experimental workflows, individual device functions, and direct actuator control, allowing existing equipment to participate at the scope its interfaces support. A level describes what the caller may invoke and change. It does not rank equipment quality, automation maturity, or scientific value. The appropriate choice depends on the experimental task and the control available.

4.1. Levels of Exposed Control

L1 provides access to an established experimental workflow as a whole. An organic synthesis workstation could, for example, add reagents, stir and heat a reaction, then cool it and collect a sample in a predefined sequence. The caller selects the recipe and permitted conditions, such as quantities,

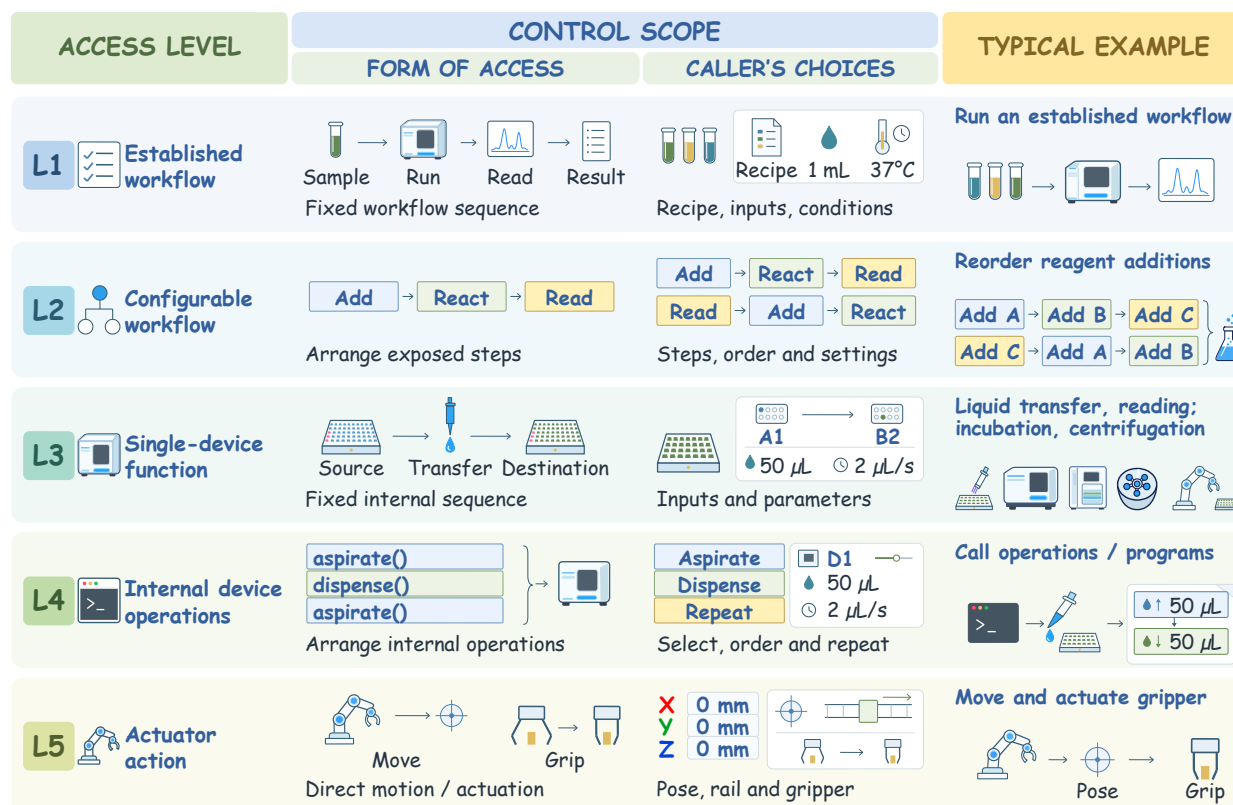


Figure 3 | **Five levels of laboratory access and caller control.** L1 and L2 concern whole experimental workflows, L3 and L4 concern individual device functions and their internal operations, and L5 exposes motion or actuation directly. The examples distinguish the execution unit invoked from the choices available to the caller. Every level may expose adjustable parameters; the classification describes control scope rather than automation maturity.

temperature, and duration, while the workstation retains the steps and their order. L2 allows the caller to select and arrange the workflow's exposed steps or modules before the local system executes it. The synthesis workstation might support staged reagent additions with mixing between them, or a different ordering of addition, reaction, and measurement steps. Both levels expose experimental choices; L2 additionally permits changes to the workflow's organization.

L3 gives access to an individual device function whose internal steps and order are predefined by the local system. The caller selects the function's inputs and exposed parameters, while the device retains responsibility for organizing those steps. For example, a liquid-transfer call can accept source and destination wells, volume, and transfer speed, while the workstation carries out aspiration and dispensing as part of the same function. The laboratory operating system can place this function at different points in a larger experimental workflow.

L4 gives the caller control over the operations inside a device function. The caller can select the exposed steps, specify their parameters, and arrange their order or repetition within the device's constraints. The local system executes this customized procedure through a supported device program or individual operation calls. For example, separately exposed aspiration and dispensing operations let a caller organize how a liquid-handling function proceeds. L2 instead concerns the arrangement of experimental stages or functions within a larger workflow, so the two levels describe configurable control at different scopes.

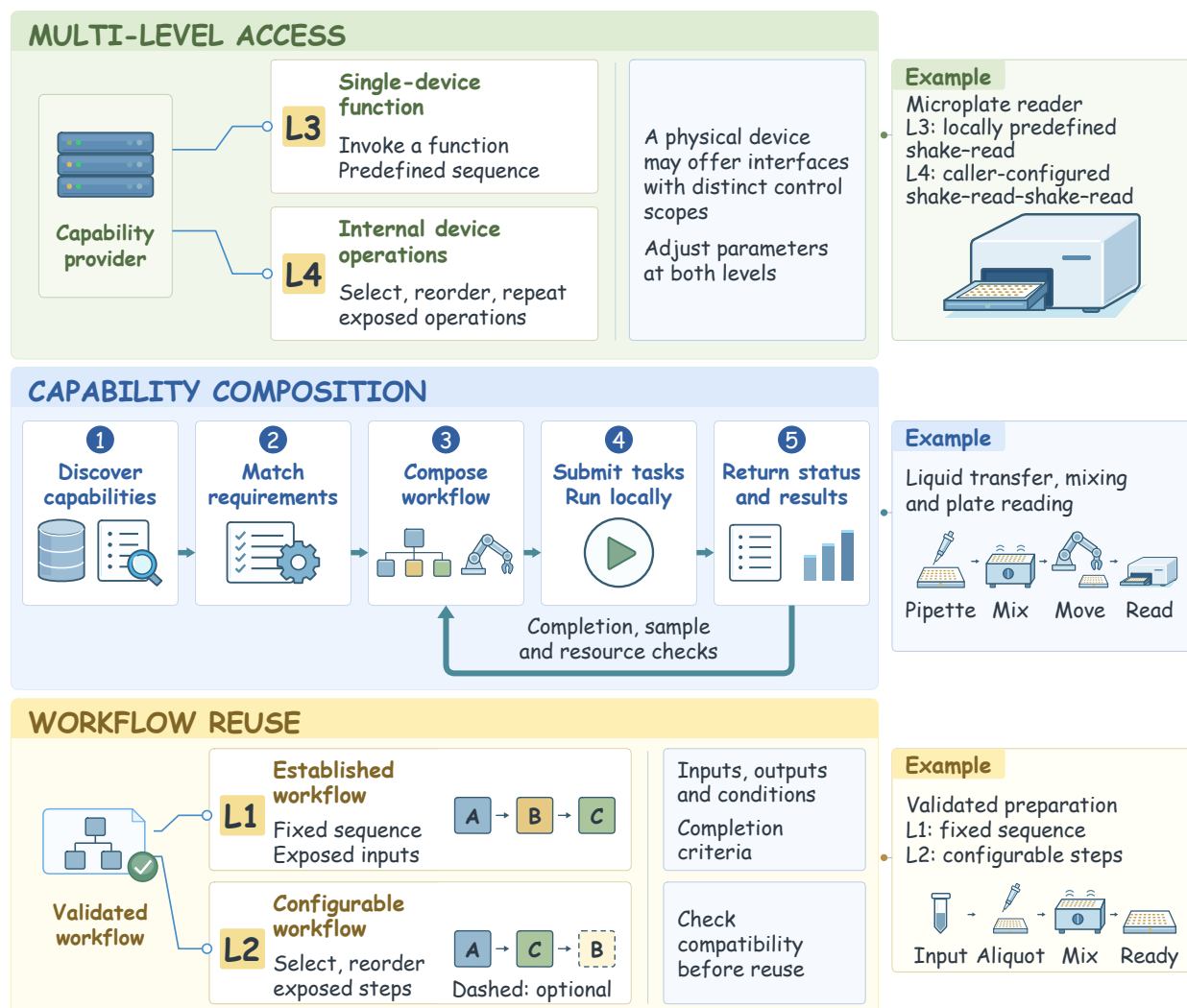


Figure 4 | **Multilevel access, capability composition, and workflow reuse.** A physical device may expose both a predefined function at L3 and caller-configurable internal operations at L4. Compatible capabilities can be discovered and combined across devices, with local execution and task feedback supporting coordination. A validated combination may then be offered as an established workflow at L1 or a configurable workflow at L2. Reuse requires compatible inputs, outputs, and operating conditions, together with declared completion criteria and checks on samples and shared resources.

L5 exposes motion or actuation directly, such as an arm pose, a rail position, or a gripper action. A caller could organize a plate pickup by specifying the approach, grasp, and lifting movements. This scope requires attention to coordinate frames, reachable space, and motion constraints. Local controllers continue to enforce the equipment's protections. The level follows the actions exposed to the caller, regardless of whether motion planning or a learned model produces those actions.

4.2. Multilevel Access and Capability Reuse

The same device may offer several levels through different interfaces (Figure 4). A microplate reader could expose a predefined *shake-read* function at L3 and also allow the caller to organize a *shake-read-shake-read* sequence at L4. Agilent's Epoch 2 documentation describes saved reading protocols and procedure editing in Gen5, including options for shaking before the first or each successive kinetic

read [46]. A researcher may prefer an established method for routine use or need to reorganize the steps for a different measurement strategy; familiarity with the instrument can also influence that choice. The classification follows the control exercised through the interface: a predefined repeated-reading method can still be offered at L3.

An electrochemical workstation could likewise offer a complete measurement method at L3 and access to its internal operations at L4. The first interface accepts settings such as potential limits and scan rate and returns a measurement curve. The second lets the caller organize control-mode selection, excitation signals, measurement-circuit switching, and data acquisition into a customized procedure. Gamry’s Explain documentation describes both standard experimental scripts and user access to these operations [47]. Each interface would have its own capability description, even though both use the same instrument.

These functions can also be combined into a reusable experimental workflow. The laboratory operating system might coordinate liquid transfer, mixing, and plate reading across devices. A laboratory could then offer a validated sequence as a single workflow, with declared inputs, conditions, and outputs. It would provide L1 access if the steps and order were fixed, or L2 access if callers could select and rearrange the exposed steps. Later experiments could reuse the workflow without organizing each device call again, while the individual functions could remain available for other combinations. A robotic plate-transfer function could participate at L3 when its local system handles the internal navigation and manipulation steps.

Reuse requires more detail than a level label. Each capability must describe its inputs, units, parameter ranges, compatible samples and containers, and required initial conditions. Effective settings and any local defaults belong in the execution record, together with the criteria and observations used to establish completion. These details help determine whether a function or workflow is suitable for another experiment. Multiple interfaces can also refer to the same physical equipment, so an individual device call must respect a workflow already using that device. Section 5 explains how task and resource records support this coordination.

5. Capability Descriptions and Runtime Coordination

5.1. Capability Descriptions and Parameter Schemas

Standardized laboratory access requires a shared description of the operations available, the inputs they accept, and the feedback they provide. We organize this information into seven categories (Table 2). MCP uses `inputSchema` to describe a tool’s accepted parameters and an optional `outputSchema` for its structured results [20]. Laboratory descriptions also need to specify parameter meanings, experimental objects, operating conditions, and completion requirements. The parameters of each operation follow its experimental requirements and the controls exposed by the device. A common description structure allows additional fields for particular instruments or disciplines.

A liquid-transfer call, for example, identifies the liquid, source and destination wells, and the volume with its unit. Its parameter schema states which values are required, their permitted ranges, and how omitted settings are handled. The description also identifies compatible containers and any required initial conditions, so the caller can judge whether the operation is suitable for the intended task. Completion requirements state how the transfer will be confirmed, while the process and result fields specify what progress and data the device can report. Safety constraints define the permitted settings and control actions.

Table 2 | Information categories for SCP-Lab capability access and task feedback. Descriptions declare the accepted inputs and available feedback; task records supply values and observations during execution.

Information category	Main fields	Purpose
Capability and level	Capability name, version, provider, and exposed access level.	Identify, discover, and reuse capabilities.
Invocation parameters	Parameters and types, units, ranges, required values, default handling, and permitted step configurations.	Specify what the caller may adjust and how to supply it.
Experimental objects	Samples, materials, containers, wells, and locations.	Identify the objects handled across devices.
Execution conditions	Equipment configuration, resource requirements, initial sample state, and container requirements.	Determine whether equipment can undertake the task.
Completion requirements	Expected state, completion criteria, and confirmation methods.	Establish whether dependent steps may proceed.
Process and results	Progress, timestamps, actual execution conditions, errors, and data.	Track execution, interpret results, and trace their history.
Safety constraints	Operating limits, invocation permissions, interlock requirements, and conditions for pause, stop, and recovery.	Define permitted operations and responses to interruptions.

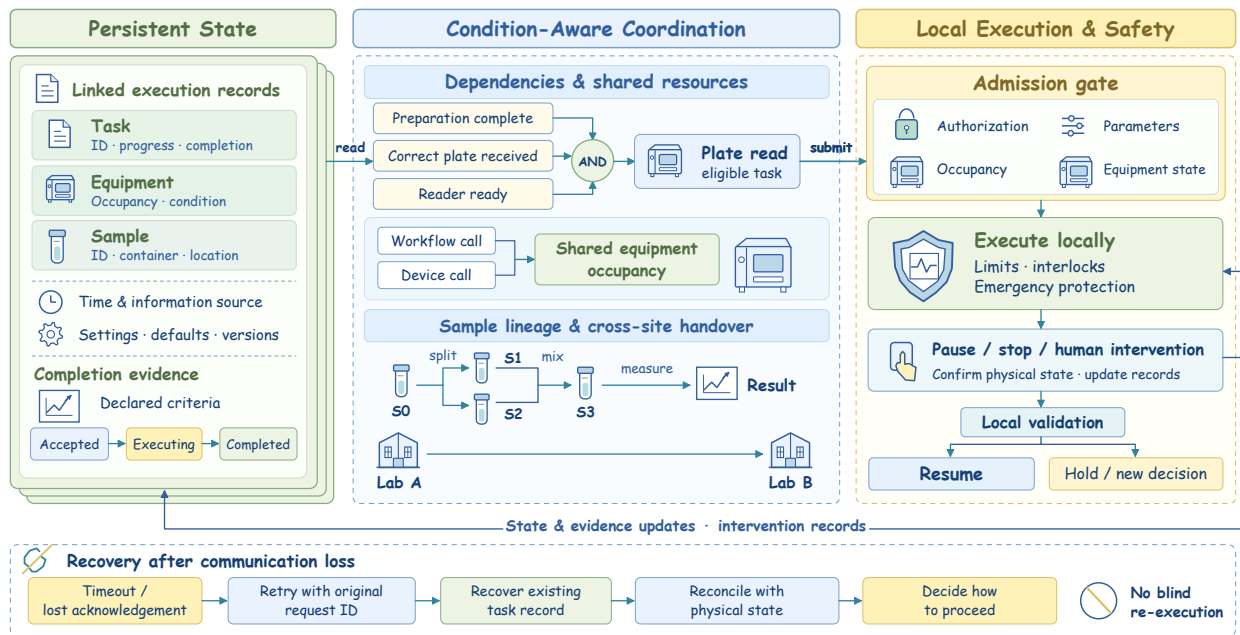


Figure 5 | **Runtime coordination and execution feedback.** Linked task, equipment, and sample records inform dependent operations, shared equipment use, and sample handovers between laboratories. Local systems check requests and enforce safety constraints; pauses and interventions update the same records. Recovery after communication loss retrieves the existing task record and checks the physical state before deciding how to proceed. Completion is assessed against declared criteria and supporting observations.

5.2. Runtime State and Communication Continuity

Each accepted task has a record that evolves as the experiment proceeds (Figure 5). We record task progress, equipment availability, and sample state together because a change in one can affect the

others. Pausing a task, for example, may leave the equipment occupied and the sample awaiting further processing. Status updates associate tasks with the relevant equipment and samples, and retain their time and information source. The laboratory operating system uses this feedback to assess current conditions and revise affected tasks. Execution records also preserve the reported settings, applied defaults, available configuration versions, and observations associated with each result. These details help researchers reconstruct how an experiment was carried out before attempting to repeat it [48]. ESCALATE, a software framework for automated chemical experimentation, similarly links experiment specifications to metadata capture and structured reporting [49].

The task record distinguishes acceptance, execution, and successful completion. Acceptance confirms that the local system has admitted the task; execution begins when local work starts, and success requires the declared completion conditions to be met. A temperature-holding task, for example, can require that a reported temperature remain within a specified range for a specified duration. Its completion record would include the temperature observations and timing used to confirm that condition. An acknowledgement that the heating command was sent establishes only that the command was issued. The same principle applies to liquid transfer: automated dispensing studies have found that delivered volumes depend on tips and operating settings [50], so a report that a dispensing sequence finished should identify whether the delivered amount was measured. The record makes the basis for declaring completion explicit. Successful execution can still yield an unfavorable scientific result, such as a correctly performed measurement that reveals failed preparation.

Task records must remain available across interruptions in communication. A liquid-transfer request may time out after dispensing has occurred, so repeating it without checking could alter the sample again. A retry should retain the original request identifier and recover the existing task within the declared scope and retention period. Changed request content under the same identifier should be rejected. The laboratory operating system can then query that task and reconcile its record with the local physical state before deciding how to proceed. Until this check is possible, the last confirmed state remains distinguishable from the current, uncertain outcome. Request deduplication alone cannot establish that every physical operation occurred exactly once.

5.3. Coordination Across Devices and Laboratories

Tasks that depend on earlier operations must wait until those operations satisfy the conditions required by the next step. Tasks without such dependencies can proceed concurrently when their sample and resource requirements permit [40]. For example, a plate-reading task can begin after sample preparation is complete, the correct plate has arrived, and the reader is ready. The protocol provides the task status, sample references, and resource information that the laboratory operating system needs to make this decision. Multiple interfaces may also share equipment: an individual reader call and a whole-workflow call must respect the same resource occupancy. The local system coordinates these requests and confirms when the equipment is ready for further work. A finished or failed task may still leave equipment requiring cleanup or cooling.

Coordinating these steps also requires knowing which material each operation receives and produces. A sample record identifies the material, its container and well, and its current location. Moving a plate updates its location; splitting a sample associates each resulting portion with its source; mixing samples records the sources of the mixture. Subsequent measurements link to the resulting sample, so the reader can trace which preparation produced the material being tested. The PROV Ontology (PROV-O) provides concepts for recording derivations and activities, while the ISA framework organizes investigations, studies, and assays [51, 52]. The FAIR principles guide how these descriptions can be made findable, accessible, interoperable, and reusable [53]. Each sample update retains its task, time, and information source, and a reported addition remains distinguishable

from a measured composition.

Cross-laboratory coordination extends these relationships to tasks and samples at different sites. Distributed experimental systems provide precedents for such collaboration [41, 42, 19]. Sites may conduct related experiments independently and return results for comparison, or contribute successive stages of preparation, processing, and measurement. Independent runs need records of method versions and actual conditions to support comparison. Successive stages also require physical handover records, including dispatch, preservation conditions, transport, and receipt of the intended sample. We retain these relationships alongside task and result records so that the operating system can follow progress across sites and respond to delays or failed handovers. Differences in calibration and local methods remain relevant when interpreting the combined evidence.

5.4. Safety and Intervention

Safety requirements govern both admission of a task and its continued execution. The capability description states operating limits, invocation permissions, interlock requirements, and supported control actions. The local execution layer verifies the caller’s authorization and checks parameters, resource occupancy, and equipment conditions before work begins. Local controllers maintain limits, interlocks, and emergency protections during execution, while reporting changes that affect whether a task can continue. Each laboratory also defines its response to communication loss. Some processes may require continued temperature or environmental control, whereas others permit a controlled shutdown. The operating system needs this policy to interpret an interruption and plan subsequent actions.

Pause, stop, and recovery must be related to their physical effects. An acknowledgement of a pause request is followed by confirmation of whether and where execution actually paused. Resumption depends on the completed steps, equipment condition, sample location, and who currently controls the equipment. Some operations can continue from a known point; others leave material that requires a new experimental decision. Human intervention updates the same record, including any changes to samples, settings, or control responsibility. The local system confirms that the revised conditions permit recovery before remote execution resumes.

6. Laboratory Integration Cases

The following cases draw on efforts to build an autonomous laboratory network spanning chemistry, materials science, and the life sciences. Figure 6 shows the experimental scenarios and their access arrangements. We first examine access to experimental workflows and individual device functions, then consider embodied execution and distributed wet-lab work. Together, these cases relate the access levels and execution requirements in Sections 4 and 5 to existing experimental practice.

Workflow access. The photoresist-synthesis apparatus at Xiamen University and the electrocatalysis equipment at Jiaying University carry out comparatively fixed experimental workflows. The caller can vary the exposed experimental parameters (e.g., reagent proportions and synthesis duration), while the existing control program retains the internal steps and their order. An L1 interface can represent each such procedure as a whole workflow, with the local system managing execution. Experiments on solid oxide functional ceramic materials at the Shanghai Institute of Ceramics, Chinese Academy of Sciences, provide another example of workflow-level access. Linking each run’s parameters, sample, and result allows a scientific agent to compare experimental conditions and use the resulting evidence to choose subsequent experiments.

Configurable workflows allow the caller to organize experimental steps. The intelligent synthesis



Figure 6 | Laboratory integration scenarios and access arrangements. The nine cases span chemistry, materials science, and the life sciences, including synthesis and materials optimization, molecular biology, drug screening, and lipid nanoparticle formulation. They illustrate access to established and configurable workflows, individual device functions, embodied operations, and distributed wet-lab execution. The central schematic relates these cases to the access levels and shared task, sample, and result records of SCP-Lab.

workstation at Fudan University allows callers to change the order of reagent addition, as well as select wells, materials, and addition amounts. The OASIS drug-screening facility at Zhejiang University organizes the use of drug plates, cell plates, and reagents through workflows and step parameters. Although these facilities serve different experimental purposes, their configurable workflows illustrate L2 access: the caller can organize the exposed steps, while each facility's existing software handles internal scheduling and execution. The protocol description must specify which steps and ordering choices are available, so that a scientific agent can determine whether a proposed experiment fits the facility's control scope. Separately callable device functions can also participate in other experimental workflows.

Device access and software integration. The molecular-biology integration project involving the Shenzhen Institutes of Advanced Technology covers liquid-handling workstations, plate readers, polymerase chain reaction (PCR) instruments, and other equipment types. Device calls follow the operating procedures of each experiment. Encapsulated liquid transfer and plate reading illustrate L3 access, while separately exposed aspiration and dispensing operations provide the finer control described by L4. Coding agents generate control programs within the available device interfaces and revise them using execution feedback. LabscripAI provides a concrete example of this approach, connecting script authoring to runtime control and bounded recovery [54]. Programs that have been

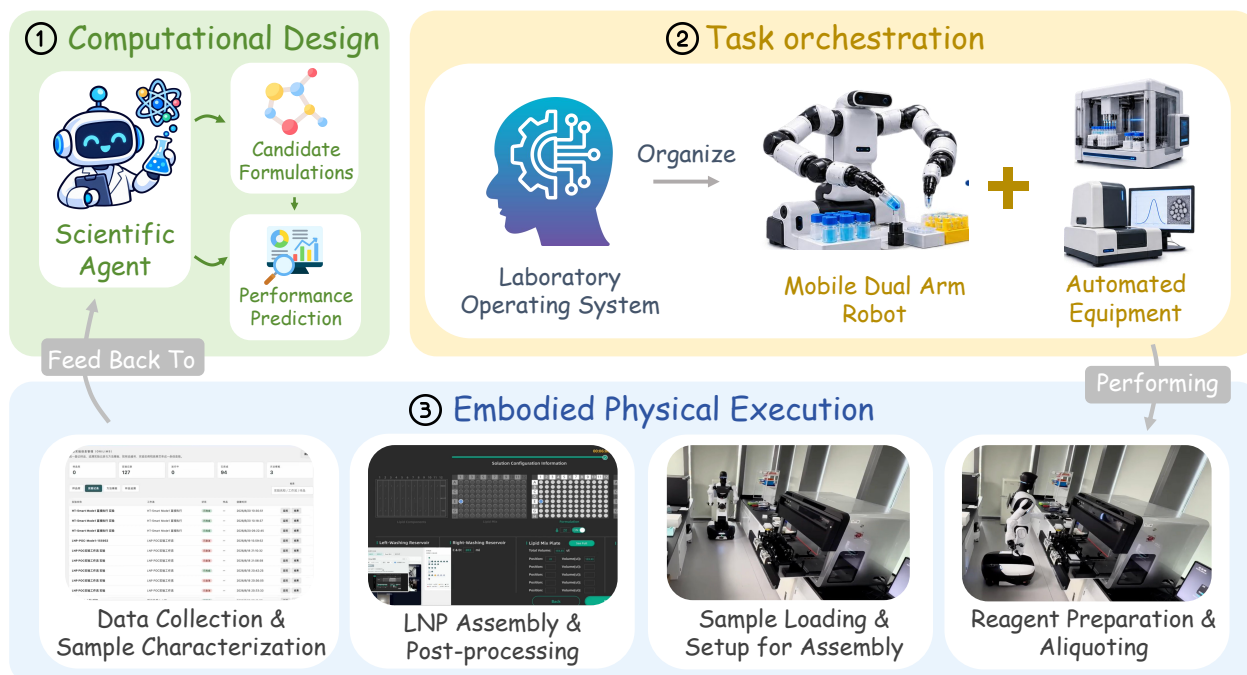


Figure 7 | **Design, execution, and feedback in LNP formulation at Lingang Laboratory.** A scientific agent proposes candidate formulations for experimental evaluation. A laboratory operating system coordinates a mobile dual-arm robot and automated equipment through reagent preparation, sample loading, LNP assembly and post-processing, and characterization. The resulting data return to the agent to guide the next round of formulation design.

verified are organized into functions or workflows with stated parameters, execution conditions, and results, allowing subsequent experiments to reuse the adaptation.

The iBiology Foundry (IB) at Zhejiang University illustrates how diverse device functions can be assembled into experimental workflows. It provides equipment for cell culture, DNA preparation and synthesis, genetic engineering, and characterization. Researchers select and combine the required devices according to their experimental proposals, then adjust workflow code and parameters for automated execution. This flexibility provides a basis for L2 access to configurable workflows, while independently callable device functions can be described at L3. The protocol description should make the available functions and permitted combinations explicit, so that a laboratory operating system can organize experiments within the platform's supported control scope. Validated combinations can subsequently be offered as reusable workflows.

Embodied execution in LNP formulation. The lipid nanoparticle (LNP) formulation workflow at Lingang Laboratory combines computational design with embodied operations and automated equipment (Figure 7). A scientific agent uses computational models to propose candidate formulations and predict their performance. A laboratory operating system then organizes the experimental tasks across a mobile dual-arm robot and the available chemical equipment. The robot performs operations such as pipetting, opening containers, and picking up or placing labware, while automated equipment carries out established preparation procedures. These complementary capabilities allow different parts of the same experiment to be assigned to the robot or to a dedicated device. Their coordination requires explicit sample references and execution feedback: the operating system needs to know which sample is ready and whether the next device can accept it. Experimental data return to the scientific agent to inform the next round of formulation choices.

Distributed protein directed evolution. The Cas12a directed-evolution workflow distributes wet-lab work across Beijing, Suzhou, and Shanghai. An upstream scientific agent uses AMix for protein design [55] and proposes candidate variants for experimental evaluation. The Beijing team performs preliminary validation and complementary functional tests. The Suzhou node undertakes molecular construction, sample preparation, and batch quality control, while the Shanghai wet-lab node uses robotics for high-throughput validation and measurement of editing efficiency. A laboratory operating system coordinates these experimental tasks and their data across sites. The main access requirement is to preserve the relationship between a candidate, its prepared samples, and the results obtained at different laboratories. Shared task and sample records allow each site to identify the material and experimental context on which its work depends, while the combined results return to the scientific agent for the next design iteration. The case therefore illustrates distributed experimental execution and the information needed to connect its separate contributions.

Across these cases, established workflows preserve local process knowledge, separately callable functions support new combinations, and embodied operations extend how equipment can participate in an experiment. Cross-site work adds the need to connect tasks, samples, and results across organizational boundaries. These practices provide a concrete basis for the protocol’s access levels and shared execution records: the interface must express both the work being delegated and the conditions needed for another operation or laboratory to continue it.

7. Discussion and Outlook

The five access levels give scientists, laboratory operators, and developers a shared vocabulary for describing experimental control. This supports a common understanding of which decisions the caller can make and which remain with the facility. An established workflow preserves local process knowledge, whereas access to device functions or internal operations enables different combinations of steps. The cases show how these interfaces can coexist, helping users choose a suitable scope of control for routine procedures and exploratory experiments.

We seek interoperability that lets independently developed scientific agents and laboratory operating systems use the same published laboratory capabilities. An operating system should be able to invoke a device function and interpret its feedback through the shared protocol, while a local adapter handles the vendor-specific commands. SiLA 2 already defines laboratory command interfaces, and LADS models devices, programs, and execution results [23, 24]. An adapter could connect these interfaces to SCP-Lab while preserving their execution semantics and describing their exposed control scope.

This interoperability offers a path for laboratories to build on one another’s integration work. A published capability can be incorporated into another laboratory’s workflow when its operating conditions meet the new task’s requirements. Verified adapters and workflow definitions can also be reused with compatible equipment. Shared cases should therefore document how capabilities were connected and what happened during execution. Records of successful runs, failed sample handovers, and uncertain execution can inform later integrations when they include the relevant observations and recovery decisions. Maintaining these descriptions and records as equipment or procedures change would help reusable capabilities accumulate across the laboratory network.

Physical completion and scientific success require distinct evidence. A completed control program does not by itself show that the intended sample state was reached, while a correctly executed experiment may still yield an unfavorable scientific result. We propose combining controller records with complementary observations, such as temperature or mass measurements and images, according to the operation being assessed. The protocol should link these observations to the task and sample,

recording their sources and times. The laboratory operating system can then assess whether to continue or request inspection when observations are missing or disagree. Appropriate measurements or assays remain necessary to evaluate the scientific outcome.

8. Conclusion

Laboratory integration brings together facilities that offer different kinds of experimental control. The cases considered here show why established workflows, configurable processes, instrument functions, and detailed operations can all be useful units of access. SCP-Lab organizes these choices through five levels of control granularity and connects them to the conditions under which a task is accepted, carried out, and interpreted. The protocol makes the division of responsibility between an experimental coordinator and a facility explicit, while preserving the sample and result relationships needed to connect their interactions.

The opportunity is to make this diversity usable within a common experimental process: to reduce repeated integration work, broaden the experiments that available resources can support, and enable collaboration across facilities and regions. Doing so calls for descriptions that explain what can be requested, observations that support the claimed outcome, and coordination that respects the current state of equipment and material. Existing standards and laboratory systems provide a foundation for developing these practices. Continued work with participating facilities can refine the proposed levels and show how capability descriptions, local execution, and scientific evidence remain connected as autonomous experimentation extends across instruments and laboratories.

Author List

Dongzhan Zhou¹, Chenxi Li¹, Zihan Guo¹, Zicai Cui¹, Yang Li^{1,6}, Minting Pan¹, Jingyuan Li^{1,6}, Bingqi Jiang¹, Chao Wu¹, Jingjing Zhang¹, Xiaosong Wang¹, Yuqiang Li¹, Jiangzhen Fu², Hao Yu², Huajie Niu³, Shuangjia Zheng⁴, Duo An⁴, Miao Tao⁴, Yeqing Yu⁴, Yali Zeng⁴, Jianjun Liu⁵, Nian Ran⁵, Tong Si⁷, Lihao Fu⁷, Yuan Gao⁷, Hao Zhou⁸, Shuyi Zhang⁸, Ziyuan Ma⁸, Wenjing Hong⁹, Ziqin Li⁹, Yikai Fang⁹, Qiaoyu Chen⁹, Yiran Huang⁹, Huajun Chen¹⁰, Kenyan Ding¹⁰, Hao Ouyang¹⁰, Ningyu Zhang¹⁰, Xujun Fu¹, Jinyu Meng¹, Tianfan Xue¹, Bin Zhao¹, Chunhua Shen¹, Lei Bai¹, Xing Xie¹, Bowen Zhou¹, Weinan Zhang^{1,6,†}

Affiliations

¹ Shanghai AI Laboratory; ² Fudan University; ³ Jiaxing University; ⁴ Lingang Laboratory; ⁵ Shanghai Institute of Ceramics, Chinese Academy of Sciences; ⁶ Shanghai Jiao Tong University; ⁷ Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences; ⁸ Tsinghua University; ⁹ Xiamen University; ¹⁰ Zhejiang University.

[†] Project Lead.

References

- [1] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021. <https://doi.org/10.1038/s41586-021-03819-2>.
- [2] Josh Abramson, Jonas Adler, Jack Dunger, Richard Evans, Tim Green, Alexander Pritzel, et al. Accurate structure prediction of biomolecular interactions with AlphaFold 3. *Nature*, 630(8016):493–500, 2024. <https://doi.org/10.1038/s41586-024-07487-w>.
- [3] Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, et al. De novo design of protein structure and function with RFdiffusion. *Nature*, 620(7976):1089–1100, 2023. <https://doi.org/10.1038/s41586-023-06415-8>.
- [4] Amil Merchant, Simon Batzner, Samuel S. Schoenholz, Muratahan Aykol, Gwooon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023. <https://doi.org/10.1038/s41586-023-06735-9>.
- [5] Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023. <https://doi.org/10.1038/s41586-023-06792-0>.
- [6] Yixiang Ruan, Chenyin Lu, Ning Xu, Yuchen He, Yixin Chen, Jian Zhang, et al. An automatic end-to-end chemical synthesis development platform powered by large language models. *Nature Communications*, 15(1):10160, 2024. <https://doi.org/10.1038/s41467-024-54457-x>.
- [7] Juraj Gottweis, Wei-Hung Weng, Alexander Daryin, Tao Tu, Anil Palepu, Petar Sirkovic, et al. Towards an AI co-scientist. <https://arxiv.org/abs/2502.18864v1>, 2025. arXiv preprint, version v1. Accessed 2026-10-01.
- [8] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. <https://arxiv.org/abs/2408.06292v1>, 2024. arXiv preprint, version v1. Accessed 2026-10-01.
- [9] Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, et al. The AI Scientist-v2: Workshop-Level Automated Scientific Discovery via Agentic Tree Search. <https://arxiv.org/abs/2504.08066v1>, 2025. arXiv preprint, version v1. Accessed 2026-10-01.
- [10] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. <https://doi.org/10.1038/s41586-023-06924-6>.
- [11] A. Christler, E. Felföldi, M. Mosor, D. Sauer, N. Walch, A. Dürauer, et al. Semi-automation of process analytics reduces operator effect. *Bioprocess and Biosystems Engineering*, 43(5):753–764, 2020. <https://doi.org/10.1007/s00449-019-02254-y>.
- [12] Ross D. King, Jem Rowland, Stephen G. Oliver, Michael Young, Wayne Aubrey, Emma Byrne, et al. The Automation of Science. *Science*, 324(5923):85–89, 2009. <https://doi.org/10.1126/science.1165620>.
- [13] Benjamin Burger, Phillip M. Maffettone, Vladimir V. Gusev, Catherine M. Aitchison, Yang Bai, Xiaoyan Wang, et al. A mobile robotic chemist. *Nature*, 583:237–241, 2020. <https://doi.org/10.1038/s41586-020-2442-2>.

- [14] B. P. MacLeod, F. G. L. Parlane, T. D. Morrissey, F. Häse, L. M. Roch, K. E. Dettelbach, et al. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 6(20):eaaz8867, 2020. <https://doi.org/10.1126/sciadv.aaz8867>.
- [15] Jacob T. Rapp, Bennett J. Bremer, and Philip A. Romero. Self-driving laboratories to autonomously navigate the protein fitness landscape. *Nature Chemical Engineering*, 1(1):97–107, 2024. <https://doi.org/10.1038/s44286-023-00002-4>.
- [16] Anne-Catherine Bédard, Andrea Adamo, Kosi C. Aroh, M. Grace Russell, Aaron A. Bedermann, Jeremy Torosian, et al. Reconfigurable system for automated optimization of diverse chemical reactions. *Science*, 361(6408):1220–1225, 2018. <https://doi.org/10.1126/science.aat0650>.
- [17] Malcolm Sim, Mohammad Ghazi Vakili, Felix Strieth-Kalthoff, Han Hao, Riley J. Hickman, Santiago Miret, et al. ChemOS 2.0: An orchestration architecture for chemical self-driving laboratories. *Matter*, 7(9):2959–2977, 2024. <https://doi.org/10.1016/j.matt.2024.04.022>.
- [18] Wenyu Zhang, Lucy Hao, Veronica Lai, Ryan Corkery, Jacob Jessiman, Jiayu Zhang, et al. IvoryOS: an interoperable web interface for orchestrating Python-based self-driving laboratories. *Nature Communications*, 16(1):5182, 2025. <https://doi.org/10.1038/s41467-025-60514-w>.
- [19] Monika Vogler, Simon Krarup Steensen, Francisco Fernando Ramírez, Leon Merker, Jonas Busk, Johan Martin Carlsson, et al. Autonomous Battery Optimization by Deploying Distributed Experiments and Simulations. *Advanced Energy Materials*, 14(46):2403263, 2024. <https://doi.org/10.1002/aenm.202403263>.
- [20] Model Context Protocol. Model Context Protocol Specification. <https://modelcontextprotocol.io/specification/2025-11-25>, 2025. Version 2025-11-25. Accessed 2026-10-01.
- [21] Yankai Jiang, Wenjie Lou, Lilong Wang, Zhenyu Tang, Shiyang Feng, Jiaxuan Lu, et al. SCP: Accelerating Discovery with a Global Web of Autonomous Scientific Agents. <https://arxiv.org/abs/2512.24189v1>, 2025. arXiv preprint, version v1. Published 2025-12-30. Accessed 2026-10-01.
- [22] Anthropic. Previewing the Model Hardware Standard. <https://www.anthropic.com/news/model-hardware-standard-research-preview>, 2026. Research preview. Published 2026-08-27. Accessed 2026-10-01.
- [23] Association Consortium Standardization in Lab Automation (SiLA). SiLA 2 Part (A) – Overview, Concepts and Core Specification. <https://sila-standard.com/wp-content/uploads/2022/03/SiLA-2-Part-A-Overview-Concepts-and-Core-Specification-v1.1.pdf>, 2022. Version 1.1. Published 2022-03-19. Accessed 2026-10-01.
- [24] OPC Foundation. OPC UA for Laboratory & Analytical Device Standard (LADS) – Part 1: Basics. <https://reference.opcfoundation.org/specs/OPC-30500-1>, 2023. Version 1.0.0. Published 2023-11-30. Accessed 2026-10-01.
- [25] Kevin Williams, Elizabeth Bilsland, Andrew Sparkes, Wayne Aubrey, Michael Young, Larisa N. Soldatova, et al. Cheaper faster drug development validated by the repositioning of drugs against neglected tropical diseases. *Journal of The Royal Society Interface*, 12(104):20141289, 2015. <https://doi.org/10.1098/rsif.2014.1289>.

- [26] Jarosław M. Granda, Liva Donina, Vincenza Dragone, De-Liang Long, and Leroy Cronin. Controlling an organic synthesis robot with machine learning to search for new reactivity. *Nature*, 559(7714):377–381, 2018. <https://doi.org/10.1038/s41586-018-0307-8>.
- [27] Connor W. Coley, Dale A. Thomas, Justin A. M. Lummiss, Jonathan N. Jaworski, Christopher P. Breen, Victor Schultz, et al. A robotic platform for flow synthesis of organic compounds informed by AI planning. *Science*, 365(6453):eaax1566, 2019. <https://doi.org/10.1126/science.aax1566>.
- [28] Robert W. Epps, Michael S. Bowen, Amanda A. Volk, Kameel Abdel-Latif, Suyong Han, Kristofer G. Reyes, et al. Artificial Chemist: An Autonomous Quantum Dot Synthesis Bot. *Advanced Materials*, 32(30):2001626, 2020. <https://doi.org/10.1002/adma.202001626>.
- [29] Amanda A. Volk, Robert W. Epps, Daniel T. Yonemoto, Benjamin S. Masters, Felix N. Castellano, Kristofer G. Reyes, et al. AlphaFlow: autonomous discovery and optimization of multi-step chemistry using a self-driven fluidic lab guided by reinforcement learning. *Nature Communications*, 14(1):1403, 2023. <https://doi.org/10.1038/s41467-023-37139-y>.
- [30] Aidan Slattery, Zhenghui Wen, Pauline Tenblad, Jesús Sanjosé-Orduna, Diego Pintossi, Tim den Hartog, et al. Automated self-optimization, intensification, and scale-up of photocatalysis in flow. *Science*, 383(6681):eadj1817, 2024. <https://doi.org/10.1126/science.adj1817>.
- [31] Nathan J. Szymanski, Bernardus Rendy, Yuxing Fei, Rishi E. Kumar, Tanjin He, David Milsted, et al. An autonomous laboratory for the accelerated synthesis of inorganic materials. *Nature*, 624(7990):86–91, 2023. <https://doi.org/10.1038/s41586-023-06734-w>.
- [32] Zhen Zhang, Zhichu Ren, Chia-Wei Hsu, Weibin Chen, Zhang-Wei Hong, Chi-Feng Lee, et al. A multimodal robotic platform for multi-element electrocatalyst discovery. *Nature*, 647(8089):390–396, 2025. <https://doi.org/10.1038/s41586-025-09640-5>.
- [33] Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6(5):525–535, 2024. <https://doi.org/10.1038/s42256-024-00832-8>.
- [34] Kyle Swanson, Wesley Wu, Nash L. Bulaong, John E. Pak, and James Zou. The Virtual Lab of AI agents designs new SARS-CoV-2 nanobodies. *Nature*, 646(8085):716–723, 2025. <https://doi.org/10.1038/s41586-025-09442-9>.
- [35] Kouros Darvish, Marta Skreta, Yuchi Zhao, Naruki Yoshikawa, Sagnik Som, Miroslav Bogdanovic, et al. ORGANA: A robotic assistant for automated chemistry experimentation and characterization. *Matter*, 8(2):101897, 2025. <https://doi.org/10.1016/j.matt.2024.10.015>.
- [36] Indrajeet Mandal, Jitendra Soni, Mohd Zaki, Morten M. Smedskjaer, Katrin Wondraczek, Lothar Wondraczek, et al. Evaluating large language model agents for automation of atomic force microscopy. *Nature Communications*, 16(1):9104, 2025. <https://doi.org/10.1038/s41467-025-64105-7>.
- [37] Sebastian Steiner, Jakob Wolf, Stefan Glatzel, Anna Andreou, Jarosław M. Granda, Graham Keenan, et al. Organic synthesis in a modular robotic system driven by a chemical programming language. *Science*, 363(6423):eaav2211, 2019. <https://doi.org/10.1126/science.aav2211>.

- [38] S. Hessam M. Mehr, Matthew Craven, Artem I. Leonov, Graham Keenan, and Leroy Cronin. A universal system for digitization and automatic execution of the chemical synthesis literature. *Science*, 370(6512):101–108, 2020. <https://doi.org/10.1126/science.abc2986>.
- [39] Loïc M. Roch, Florian Häse, Christoph Kreisbeck, Teresa Tamayo-Mendoza, Lars P. E. Yunker, Jason E. Hein, et al. ChemOS: An orchestration software to democratize autonomous discovery. *PLOS ONE*, 15(4):e0229862, 2020. <https://doi.org/10.1371/journal.pone.0229862>.
- [40] Dan Guevarra, Kevin Kan, Yungchieh Lai, Ryan J. R. Jones, Lan Zhou, Phillip Donnelly, et al. Orchestrating nimble experiments across interconnected labs. *Digital Discovery*, 2(6):1806–1812, 2023. <https://doi.org/10.1039/d3dd00166k>.
- [41] Jiaru Bai, Sebastian Mosbach, Connor J. Taylor, Dogancan Karan, Kok Foong Lee, Simon D. Rihm, et al. A dynamic knowledge graph approach to distributed self-driving laboratories. *Nature Communications*, 15:462, 2024. <https://doi.org/10.1038/s41467-023-44599-9>.
- [42] Felix Strieth-Kalthoff, Han Hao, Vandana Rathore, Joshua Derasp, Théophile Gaudin, Nicholas H. Angello, et al. Delocalized, asynchronous, closed-loop discovery of organic laser emitters. *Science*, 384(6697):eadk9227, 2024. <https://doi.org/10.1126/science.adk9227>.
- [43] Jing Gao, Junhan Chang, Haohui Que, Yanfei Xiong, Shixiang Zhang, Xianwei Qi, et al. Uni-LabOS: An AI-Native Operating System for Autonomous Laboratories. <https://arxiv.org/abs/2512.21766v1>, 2025. arXiv preprint, version v1. Accessed 2026-10-01.
- [44] Model Context Protocol. Model Context Protocol Specification: Tasks. <https://modelcontextprotocol.io/specification/2025-11-25/basic/utilities/tasks>, 2025. Version 2025-11-25. Accessed 2026-10-01.
- [45] Zekun Ren, Hongzhao Tan, Jiaen Yee, and Kedar Hippalgaonkar. PUDA: An AI-Native Hardware Harness for Self-Driving Laboratories. <https://arxiv.org/abs/2607.26464v1>, 2026. arXiv preprint, version v1. Accessed 2026-10-01.
- [46] Agilent Technologies, Inc. Agilent BioTek Epoch 2 Absorbance Microplate Spectrophotometer: Instructions for Use. https://www.agilent.com/cs/library/packageinsert/public/Epoch2_IFU_1771011IAW.C.pdf, 2022. Document 1771011I, Revision C, May 2022. Accessed 2026-10-01.
- [47] Gamry Instruments, Inc. Customizing Electrochemical Experiments with the Explain Scripting Language. <https://www.gamry.com/assets/Uploads/Customizing-Electrochemical-Experiments-with-the-Explain-rev-2.pdf>, 2015. Application note, Revision 2.0, March 20, 2015. Accessed 2026-10-01.
- [48] Timothy M Errington, Alexandria Denis, Nicole Perfito, Elizabeth Iorns, and Brian A Nosek. Challenges for assessing replicability in preclinical cancer biology. *eLife*, 10:e67995, 2021. <https://doi.org/10.7554/elife.67995>.
- [49] Ian M. Pendleton, Gary Cattabriga, Zhi Li, Mansoor Ani Najeeb, Sorelle A. Friedler, Alexander J. Norquist, et al. Experiment Specification, Capture and Laboratory Automation Technology (ES-CALATE): a software pipeline for automated chemical experimentation and data management. *MRS Communications*, 9(3):846–859, 2019. <https://doi.org/10.1557/mrc.2019.72>.
- [50] Bala Mounika Reddi, Apoorva Saxena, Ronaldo L. Magtoto, Luis G. Giménez-Lirola, Phillip C. Gauger, and Rahul Kumar Nelli. A practical and affordable photometric method for routine low volume verification of fixed-head automated liquid handling systems. *SLAS Technology*, 39:100442, 2026. <https://doi.org/10.1016/j.slast.2026.100442>.

-
- [51] World Wide Web Consortium. PROV-O: The PROV Ontology. <https://www.w3.org/TR/2013/REC-prov-o-20130430/>, 2013. W3C Recommendation, 30 April 2013. Accessed 2026-10-01.
- [52] Susanna-Assunta Sansone, Philippe Rocca-Serra, Dawn Field, Eamonn Maguire, Chris Taylor, Oliver Hofmann, et al. Toward interoperable bioscience data. *Nature Genetics*, 44(2):121–126, 2012. <https://doi.org/10.1038/ng.1054>.
- [53] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3(1):160018, 2016. <https://doi.org/10.1038/sdata.2016.18>.
- [54] Y. Gao, L. Fu, Y. Liu, Y. Luo, J. Zhang, W. Li, et al. Execution-aware agent harness for accessible and responsible synthetic biology automation. <https://www.biorxiv.org/content/10.1101/2025.09.30.679666v3>, 2026. bioRxiv preprint, version v3, 24 September 2026. Accessed 2026-10-01.
- [55] Keyue Qiu, Yixin Wu, Lihao Wang, Yawen Ouyang, Jixiang Yu, Zihan Zhou, et al. AMix-2: Establishing Protein as a Native Modality in Large Language Models. <https://arxiv.org/abs/2605.30963v1>, 2026. arXiv preprint, version v1. Accessed 2026-10-01.

A. Illustrative Protocol Examples

These examples illustrate three key aspects of SCP-Lab: capability discovery and device selection, control of device functions at different access levels, and runtime task management. The runtime example focuses on task identity, completion evidence, and interruption. Algorithms and Python-style fragments use illustrative names and fields. Throughout, r denotes a request, R its task record with state and execution evidence, π an ordered operation sequence, and $\perp$ no retained record.

A.1. Capability Discovery and Device Selection

Algorithm A.1. Select a device and submit a temperature-holding task

Input: Temperature-holding request r with sample, settings, and request identifier

Output: Accepted task record R , or a rejection reason

```

1:  $C_{all} \leftarrow \text{DiscoverCapabilities}()$                                 ▶ Published capability offers and their devices
2:  $C_{fit} \leftarrow \text{FilterCompatible}(C_{all}, r)$                         ▶ Match the sample and requested settings
3: if  $C_{fit} = \emptyset$  then return  $\text{Reject}(\text{"unsupported task"})$ 
4:  $c \leftarrow \text{Select}(C_{fit})$                                           ▶  $c$ : selected capability and its device
5:  $R \leftarrow \text{SubmitTask}(c, r)$                                        ▶ Device validates, admits, and creates the record
6: return  $R$ 

```

An agent plans to incubate plate P at 37 °C for 30 minutes before a measurement. It reads the available devices' published capabilities and selects a device that accepts the plate and supports the requested temperature and duration. The agent then sends a `temperature_hold` request with the plate identifier and these settings, as shown on the left of Code A.1. The device's local software checks the request and returns a task identifier upon acceptance. During execution, it reports the applied settings and temperature readings under that task identifier, as illustrated on the right.

Code A.1. Requested settings and locally reported execution evidence.

Task request

```

request = {
  "request_id": "hold-017",
  "capability": "temperature_hold",
  "sample_id": "plate-P",
  "parameters": {
    "temperature_degC": 37,
    "duration_min": 30,
  },
}

```

Execution record

```

record = {
  "task_id": "task-042",
  "request_id": "hold-017",
  "state": "running",
  "effective_parameters":
    applied_parameters,
  "observations": [
    {
      "temperature_degC": T_observed,
      "observed_at": timestamp,
      "source": sensor_id,
    },
  ],
}

```

`applied_parameters` contains the locally applied settings. `T_observed` is one temperature reading; `timestamp` and `sensor_id` identify its time and source. This record illustrates one observation during execution.

In Algorithm A.1, C_{all} contains the discovered offers, while C_{fit} contains those compatible with the request. `SubmitTask` groups the device-side checks: authorization, units, defaults, parameter limits, sample compatibility, and resource availability. Accepted work receives a task record retaining the original request, capability version, and effective settings; rejected work receives a reason. Acceptance precedes execution. Completion requires a temperature trace supporting the 30-minute hold within the declared tolerance. The agent uses that evidence and current plate and equipment conditions to schedule the measurement.

Algorithm A.2. Execute within the control scope offered by the interface

Input: L3 or L4 capability c ; transfer request r ; agent-selected sequence π_A at L4

Output: Task record R with feedback and outcome, or a rejection reason

```

1: if  $c.level = L3$  then                                ▶  $c.level$ : published access level
2:    $\pi \leftarrow \text{LocalExpand}(c, r)$                     ▶ Device defines the internal procedure
3: else
4:    $\pi \leftarrow \pi_A$                                 ▶ L4: agent chooses the order and repetitions
5: end if
6:  $R \leftarrow \text{ValidateAndAdmit}(c, \pi)$                 ▶ Check the plan and reserve resources
7: if  $R.state \neq \text{ACCEPTED}$  then return  $R$ 
8: for each operation  $o$  in  $\pi$ , in order do
9:    $R \leftarrow \text{ExecuteAndRecord}(R, o)$                 ▶ Retain feedback and the last confirmed operation
10:  if  $\neg \text{LastStepComplete}(R)$  then break
11: end for
12: return  $\text{AssessOutcome}(R, c.completion)$             ▶ Use the declared completion condition

```

A.2. Control Granularity at L3 and L4

For a liquid-handling task, an agent needs to transfer 50 μL from well A1 to well B1. Through an L3 interface, it calls `transfer` with the source, destination, and volume, and the liquid handler carries out its predefined procedure. Through an L4 interface, the agent can choose and arrange individual operations. Code A.2 contrasts the function call with a sequence in which the agent picks up a tip, aspirates, dispenses, mixes the destination three times, and drops the tip. Volumes are expressed in microliters, as declared by the corresponding capability.

Code A.2. Agent control through a function call or a sequence of operations.

L3 Predefined function

```

transfer(
  source="A1",
  destination="B1",
  volume_uL=50,
)

```

L4 Configurable operations

```

pick_up_tip()
aspirate("A1", 50)
dispense("B1", 50)
mix("B1", 50, repeat=3)
drop_tip()

```

In Algorithm A.2, π is the plan selected for execution. At L3, `LocalExpand` supplies the device's predefined procedure, which may contain repeated steps. At L4, π is the agent's sequence π_A , such as the one in Code A.2. `ValidateAndAdmit` checks operations, order, parameter limits, sample compatibility, and local resource conditions. If these requirements are met, the device accepts the plan and proceeds to execution; otherwise, it returns a record identifying the unmet requirement. `ExecuteAndRecord` returns when an operation completes, is interrupted, or its outcome becomes uncertain; an incomplete operation prevents further steps. The final assessment follows the evidence checks illustrated in Algorithm A.3. The same device can expose both interfaces.

The agent can also combine a transfer on a liquid handler, a hold in an incubator, and a measurement on a plate reader. Invoking these L3 functions as part of an experimental plan preserves each function's access level. If a laboratory publishes the entire workflow as one capability, a fixed sequence is L1, while allowing the agent to select and arrange its exposed stages makes it L2. L4 concerns the operations within a device function, such as the liquid-handling sequence above; L5 concerns directly exposed motion or actuation.

Algorithm A.3. Recover task identity and assess the reported outcome

Input: Original request r with unchanged identifier and content; completion condition Q

Output: Refreshed task record R with reported outcome, or a reconciliation reason

```

1:  $R \leftarrow \text{RecoverTask}(r)$  ▷ Look up the original task record
2: if  $R = \perp$  then return "reconciliation required"
3:  $R \leftarrow \text{RefreshRecord}(R)$  ▷ Feedback, last confirmed operation, current conditions
4: if  $\text{InProgress}(R)$  then return  $R$  ▷ Preserve a confirmed active state
5: if  $\text{EvidenceSupports}(R, Q)$  then
6:    $R.\text{state} \leftarrow \text{SUCCEEDED}$ 
7: else if  $\text{FailureConfirmed}(R)$  then
8:    $R.\text{state} \leftarrow \text{FAILED}$ 
9: else if  $\text{StopConfirmed}(R)$  then
10:   $R.\text{state} \leftarrow \text{STOPPED}$ 
11: else
12:   $R.\text{state} \leftarrow \text{UNRESOLVED}$ 
13: end if
14: return  $R$ 

```

A.3. Runtime Task Management

Suppose the agent sends the liquid-transfer request, but its connection drops before the acceptance response arrives. The liquid handler may already be transferring the liquid. The agent resends the unchanged request with its original identifier. The local software uses that identifier to find the previously accepted task and returns its existing task identifier, allowing the agent to query whether the transfer is still running, has completed, or was interrupted. In Code A.3, `transfer_request` denotes that unchanged request, including its identifier and transfer parameters. The fragment illustrates how a retry refers to the same task, followed by a cancellation request and a query for the outcome.

Code A.3. Retain task identity across retries and query the effect of an intervention.

```

first = submit(transfer_request)
retry = submit(transfer_request) # Same identifier and content
assert retry.task_id == first.task_id

cancel(first.task_id) # Request a stop
record = query(first.task_id) # Inspect state and supporting evidence

```

If the agent requests a stop during dispensing, the acknowledgement confirms receipt of that request. The transfer may finish before the stop takes effect; an interrupted transfer may leave liquid in the tip and require local recovery. In Algorithm A.3, `RefreshRecord` retains observations, the last confirmed operation, and current sample and device conditions. `EvidenceSupports` checks those observations against Q , while `StopConfirmed` requires feedback that execution actually stopped. The agent uses the record to decide what work can proceed. Equipment release separately requires confirmed readiness for further use.

The service specifies where request identifiers must be unique and how long it retains their association with tasks. While that association is retained, an unchanged retry within the same identifier scope returns the original task identifier without creating a second task. This does not guarantee exactly-once physical execution. Reusing an identifier with changed content is rejected; a missing association requires reconciliation before new execution when earlier acceptance remains possible. Recovery checks retained records against device feedback, preserving uncertainty when evidence is missing or stale. The agent can then establish the conditions needed for further work. A confirmed task outcome records physical execution; interpreting its experimental significance remains a separate step.