Community Infrastructure Use Cases

July 17, 2019

Version 2.1

This document outlines the typical needs of the "Community Infrastructure" area. The public research computing environment should be open and extensible. In order to participate in this environment, community members (application developers, service providers, campus IT administrators) must be able to access details about the system's design, implementation status, and the driving user needs.

Note: In these use cases, the terms "service" and "service provider" are used broadly. Any of the following would be examples of services: a computer system, a website or web-based application, a computer system with a specific software application installed for use, a high-capacity data storage device, a long-term data archiving service, a virtual machine (IaaS) host, a software repository, or an identity provider.

CI-01 through CI-03 are generic use cases that describe generalized data access/publish/change features. The rest of the use cases are specific community needs that make use of the three generalized features. CI-01 through CI-03 overlap significantly with the enabling functions CAN-08, CAN-09, CAN-11, and CAN-12, and will be removed in subsequent versions of the Community Infrastructure use cases.

CI-01: Access system information

A **community member** needs to access specific information about the public research computing environment. Several specific examples of this general need follow.

a. A **researcher** has been invited to participate in a use case review.
b. An **application developer** or a **service provider** needs to find underserved use cases in order to offer something of value to the community.
c. A researcher, an application developer, or a service provider needs to know which use cases are currently being worked on in order to avoid duplicating effort.
d. A researcher, a service provider, an application developer, or a **campus IT administrator** needs to find software for a particular use.
e. A researcher needs to find a service provider who offers a particular service.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member opens a web browser and navigates to an information website.
2. Then, the community member navigates the website to locate the system information section.
3. Finally, the community member locates the specific system details he/she needs.

We'll take any solution, as long as the following are true.

1. The community member shouldn't need to be registered or need to login to the information website to access the necessary information.
2. No client software, other than a standard web browser, should be needed to access or use the necessary information.
3. The community member shouldn't be able to change the system information he/she is accessing.

CI-02: Manually publish system information

A **community member** needs to update the public record of specific system information so others can find and use the information. Several specific examples of this general need follow.

a. An **community member** needs to revise/update the known use cases.
b. A community member needs to update the implementation status of one or more use cases.
c. A community member needs to publish a plan to change an implementation of one or more use cases.
d. A **software provider** needs to add/update information about their software.
e. A **service provider** needs to add/update availability of a service.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member opens a web browser and navigates to an information website.
2. Then, the community member navigates the website to locate the system information section.
3. Then, the community member locates the specific system details he/she needs to change (or the place he/she needs to add new information).
4. Finally, the community member makes the necessary changes.

It'll always be like this except, in Step 4, when the community member hasn't already authenticated with the system, or if the latest authentication has expired. In that case, the community member will need to authenticate before he/she will be able to change any information.

We'll take any solution, as long as the following are true.

1. No client software, other than a standard web browser, should be needed to access or change the necessary information.
2. The operators of the information website must be able to authorize specific individuals and/or groups to change specific types of information. For example: a specific XSEDE staff member can change use cases; a specific software author can change details about his/her software products; a specific service provider staff member can change details about his/her services.
3. The information website must use a community-wide login service as described in use case CAN-6.

CI-03: Automate changes to system information

A **community member** needs to automate updates to system information so that updated information about one or more components can become available without human action.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member registers with an information service and obtains a credential for updating system information.
2. Then, the community member downloads and installs a tool for pushing status updates to the information service.
3. Then, the community member configures the information update tool with the credential provided in Step 1, a means to gather information about the component(s), and a mechanism for periodically executing the application.
4. Finally, the community member checks the information website (as described in use case CI-1) to confirm that the information about the component(s) is correct and current.

It'll always be like this except in rare instances when the community member chooses to build the status update mechanism directly into a component using a software application programmer interface (API). In that case, Step 2 either won't happen at all or will be replaced by downloading a software

development kit (SDK) and Step 3 will be configuring the component to update the information service using the API.

We'll take any solution, as long as the following are true.

1. In its simplest form, the registration mechanism in Step 1 should allow registration via an email request.
2. The information update tool in Step 2 must work on a wide variety of Linux/Unix distributions and hardware architectures.
3. The information update tool in Step 3 should include configuration examples for the most commonly used (and commonly provided) services.
4. The credential provided in Step 1 should only allow the community member to update the system information for his/her component(s).

CI-04: Publish current and desired system capabilities, their availability, and their implementation status

A **community member** needs to create a public record of system capabilities (aka "use cases") that are needed by members of the research community.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member opens a web browser and logs in to an information website.
2. Then, the staff member selects the "new use case" option (specific wording may vary) and receives a form to complete.
3. Then, the staff member fills out and submits the form.
4. Finally, the staff member sees the new use case listed in appropriate views within the information website.

We'll accept any solution as long as the following are true.

1. In Step 1, the information website must use a community-wide login service as described in use case CAN-6.
2. In Step 3, the form should allow the staff member to enter a link (web address) for the full capability description and an implementation status field to describe (or cross-reference) the current implementation status of the capability.

CI-05: Discover current and desired system capabilities, their availability, and their implementation status

Researchers, educators, science gateway developers, application developers, campus IT administrators, campus champions, and service providers (hereafter referred to as **"community members"**) need to

discover system capabilities (aka “use cases”) that EITHER: (a) are currently offered by the system and that might enable their work, OR: (b) are currently needed and that they might be able to supply.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member opens a web browser and visits the information website.
2. Then, the community member browses or searches the site for relevant capabilities.
3. Whenever the community member finds a relevant capability, he is able to follow links and cross-references to find a full description, its implementation status, and its current availability within the system.

We’ll accept any solution as long as the following are true.

1. User authentication should not be required for any of the steps.
2. In Step 2, the site should provide both category browsing and keyword search. Browsing should be possible by function (general to specific) and by intended user type (e.g., “educators”). It should be possible to filter searches by implementation status (e.g., “not implemented”).

CI-06: Rate the priority or quality of a community need or contribution

A user representative, service provider, or other community member (hereafter referred to as **“community member”**) needs to be able to rate the priority of a community need (a capability that isn’t currently available) or the quality of a contribution (an implemented capability) relative to her work.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member receives an email message that includes a web link to a rating interface.
2. Then, the community member opens the link in a web browser and logs into the rating interface.
3. Then, the community member uses the interface to prepare her ratings.
4. Then, the community member uses the interface to submit her ratings.
5. Finally, when the rating period is over, the community member’s ratings will be included in the results.

We’ll accept any solution as long as the following are true.

1. In Step 3, the information website must use a community-wide login service as described in use case CAN-6.
2. In Step 3, the interface should provide all of the instructions necessary to complete the ratings.
3. In Step 4, once the community member has submitted her ratings, the data is immediately available to rating administrators (but to no one else).
4. In Step 5, the community member’s identity must not be included in any public results.

CI-07: Discuss a community activity

Community members need to discuss the priority, plan, design, or final product of an activity affecting the capabilities available to the community.

In most cases, the **community members** want to experience it as follows.

1. First, the community member receives an email invitation containing the web address of the forum.
2. Then, the community member opens the link in a web browser and logs into the discussion forum.
3. Then, the community member reviews discussion topics and optionally responds to those topics.
4. Then, the community member adds his own feedback as new topics. These topics become available for other participants to see and respond to.
5. Finally, when the discussion period is over, the record of the discussion (all topics and responses) is available for public review.

We'll accept any solution as long as the following are true.

1. In Step 2, the information website must use a community-wide login service as described in use case CAN-6.
2. In Steps 3, 4, and 5, participants in the discussion must have the option of editing or deleting their own responses.

CI-08: Conduct an engineering review of a community contribution

A **community member** needs to facilitate an engineering review of the design and/or security issues for a proposed contribution to the system.

In most cases, the community member wants to experience it as follows.

1. First, the community member opens a web browser and logs into the engineering review website.
2. Then, the community member selects the "new review" option (specific wording may vary) and is presented with a form for entering the information about the review.
3. Then, the community member completes and submits the form. The community member obtains a web address for the review and sends it via email to other community members to invite their participation.
4. Then, each community member who receives an email invitation to the review opens the web address in a web browser and logs in to the review interface.
5. Then, the community member uses the review interface to view the materials to be reviewed and participate in related discussion forums (see CI-07) or ratings (see CI-06).

6. Then, if a community member is satisfied with the materials under review, she selects the "approve" option to sign off on the review.
7. Finally, once the review period is complete, all of the discussions and/or rating data is available and the community member who initiated the review can close it.

We'll accept any solution as long as the following are true.

1. In Steps 1 and 4, tthe information website must use a community-wide login service as described in use case CAN-6.
2. In Step 2, the form for entering review information must allow the community member to link to relevant documents, to name required and optional reviewers, to list specific review criteria, and to name other community members who have special roles in the review.
3. In Step 6, an electronic record of all sign-offs (reviewer & date) is maintained indefinitely.
4. In Step 7, appropriate records of each review are maintained on the review website indefinitely and can be viewed by community members.

CI-09: Discover and inspect system capabilities currently under development

Service providers, application and science gateway developers, researchers, and educators (hereafter referred to as **"community members"**) need to discover and inspect any new and enhanced capabilities currently being worked on so that they can prepare to leverage them or avoid duplicating them.

In most cases, the **community member** wants to experience it as follows.

1. First, the community member opens a web browser and visits the information website.
2. Then, the community member navigates the site to locate work in progress. The site displays all known activities related to each system capability (aka "use case") with appropriate links to details.

We'll accept any solution as long as the following are true.

1. None of these steps should require user registration or logging in.
2. In Step 2, the information website should offer search, filtering, and sorting/organizing options for the list of activities.
3. In Step 2, any relevant rating data (see CI-06) should be included as context for activities.

CI-10: Discover and review engineering documents for a specific capability or component

A service provider, system operator, or system integrator (hereafter referred to as **"operator"**) needs to discover and review the documentation and any engineering process artifacts (e.g., deployment plans) available for a specific system component.

In most cases, the **operator** wants to experience it as follows.

1. First, the operator opens a web browser and visits the information website.
2. Then, the operator navigates the site to locate information about installable components. The site displays the known components and related documentation and engineering artifacts with appropriate links to detailed information.

We'll accept any solution as long as none of these steps require user registration or logging in.

CI-11: Manage the source code, documentation, and installable packages for a capability

A staff member, system integrator, or software provider (hereafter referred to as **"maintainer"**) needs to manage the source code, documentation, and installable packages used to implement one or more system capabilities (aka "use cases").

In most cases, the **maintainer** wants to experience it as follows.

1. First, the maintainer requests a place in the system's source code repository for his specific component(s). He receives read and write access to a portion of the repository and information about the repository's use and conventions.
2. Then, the maintainer populates the repository with any items (source code, documentation, packages, configuration files, etc.) that are needed to support the relevant system capabilities.
3. Then, the maintainer provides references to items in the repository in plans to support specific capabilities/use cases and others retrieve these items as plans are reviewed and carried out.
4. Then, the maintainer manages changes to the repository's contents over time as improvements are made or user needs change.
5. Finally, if the components are no longer needed, the maintainer removes the items from the repository.

We'll accept any solution as long as the following are true.

1. In Step 1, the system repository must be widely used in the software development and system configuration management community, have broad support from many sources, and have client tools available for free use.
2. In Step 3, references to items in the repository must be able to specify versions of the items at a given point in time, not just the most recent version. (See Step 4.)
3. In Step 4, the repository must keep a revision history for all items in the repository allowing previous versions to be retrieved if necessary and all changes to be inspected.

History

	Version	Date	Changes	Author
Entire Document	1.0	11/14/2016	First version, described the general use cases (1-3)	Liming, Navarro
Use cases 01-03	1.1	4/20/2017	Revised use cases 1-3 to be more concrete and actionable	Liming, Navarro
Use cases 04-11	2.0	8/17/2018	Added specific use cases based on how use cases 1-3 have been used in practice during the past year	Liming
Entire document	2.1	7/17/2019	Removed unnecessary XSEDE terminology	Liming