XSEDE Canonical Use Case 2:

Managed File Transfer

March 20, 2014

Version 1.2

XSEDE
Extreme Science and Engineering
Discovery Environment

Table of Contents

A. Document History

Overall Document Authors:

Ian Foster
The University of Chicago and Argonne National Laboratory
Argonne, IL   60439
foster@anl.gov

Morris Riedel Jülich Supercomputing Centre
Forschungszentrum Jülich GmbH
D-52425 Jülich
Germany

Felix Bachmann
Carnegie Mellon University
4500 5th Avenue
Pittsburgh, PA   15213
fb@sei.cmu.edu

Andrew Grimshaw
University of Virginia
PO Box 400740
Charlottesville VA 22904
grimshaw@virginia.edu

David Lifka
Cornell University
512 Frank H. T. Rhodes Hall
Ithaca, NY   14853
lifka@cac.cornell.edu

	Version	Date	Changes	Author
First use case draft	0.1	3/21/2013	Document created	Foster, Grimshaw, Hossain, Lifka,

				Riedel, Tuecke
Formatted draft	0.2	04/08/13	Applied standard formatting	Hossain
Revised draft	0.2	04/25/13	Cleaned up draft; ready for archiving	Hossain
Revised draft	1.0	08/29/13	Separated from UCCAN 1&3; ready to be archived	Brown, Hossain
Revised 1.0	1.1	11/8/13	Revisions based on review by stakeholders (C. Jordan, J.P. Navarro, S. Sakai, S. Smallen, D. Simmel, V. Hazelwood)	Liming
Revised title	1.2	3/20/14	Changed title to "Managed File Transfer"	Liming

B. Document Scope

This document is both a user-facing document (publically accessible) and an internal working document intended to define user needs and use cases that fall within the overall activities of XSEDE. The definition of use cases is based on a template from Malan and Bredemeyer[1]. In general it is in keeping with the approaches and philosophy outlined in "Software architecture in practice."[2]

This document is one component of a process that generates at least the following documents, some of which are user-facing, some are as of now intended to be internal working documents:

- ***This document*** - A description of use cases [User facing]
- A set of level 3 decomposition documents, which include:
    - Quality Attributes descriptions
    - Connections diagram in UML

The use cases are presented here using the following format, derived from the Malan and Bredemeyer white paper[1] as follows:

Use Case	Use case identifier and reference number and modification history
Description	Goal to be achieved by use case and sources for requirement
References	References and citations relevant to use case
Actors	List of actors involved in use case
Prerequisites (Dependencies) & Assumptions	Conditions that must be true for use case to be possible Conditions that must be true for use case to terminate successfully
Steps	Interactions between actors and system that are necessary to achieve goal
Variations (optional)	Any variations in the steps of a use case
Quality Attributes	
Non-functional (optional)	List of non-functional requirements that the use case must meet
Issues	List of issues that remain to be resolved

[1] Malan, R., and D. Bredemeyer. 2001. Functional requirements and use cases. *www.bredemeyer.com/pdf_files/functreq.pdf*
[2] Bass, L., P Paul Clements, and Rick Kazman

C. Glossary

Please refer to the *Glossary of Terms for Canonical Use Cases* document.

D. Canonical Use Case 2

Use Case UC CAN 2	Managed File Transfer
Description	The user instructs the system to transfer one or more files and/or directories from one file system to another and the system accomplishes it.
References	The following documents are available via the XSEDE Use Case Registry, https://software.xsede.org/registry-dev/index.php. *XSEDE Campus Bridging Use Cases*, v1.5, 16 March 2012. *XSEDE Federation and Interoperation Use Cases*, v0.3, 22 January 2013. *XSEDE Data Analytics Use Cases*, v0.3, 14 June 2013. *XSEDE Data Management Use Cases*, v1.5, 24 April 2013. *Visualization Use Cases*, v1.3, 5 March 2013. *XSEDE Scientific Workflow Use Cases*, v0.3, 3 April 2013. *XSEDE Science Gateway Use Cases*, v0.4, 25 October 2012.
Actors	User: The user initiates the file transfer activity and presents a valid file transfer request. The user may be interacting directly with a command line interface, a Web browser, or an application such as a gateway, shell script, or Java, C, or Python program.
Prerequisites (Dependencies) & Assumptions	a) The user knows the address of the file transfer service. b) The user is properly authenticated. c) The user has generated a file transfer request in the appropriate format, specifying the file(s) and/or directory(s) that are to be transferred, and the source and destination file systems. d) The source file(s) and directory(s) named in the request exist and the user is authorized to read them. e) There is sufficient space for those file(s) and directory(s) on the destination file system, and the user is authorized to write at the specified destination location. f) The file transfer service, source file system, destination file system, and intervening network do not fail during execution of the file transfer.
Steps	1. The user sends a file transfer request to the file transfer service and receives some form of request identifier in return. 2. (optional) The user polls the file transfer service periodically to determine the status of the transfer: for example, the amount of data and number of files that have been transferred successfully, and whether the transfer has started, completed, or failed. 3. (optional) The user registers with the file transfer service to receive

	notifications of file transfer state change. 4. The file transfer service begins execution of the transfer. 5. (optional) The user cancels the transfer. This could occur either before or after step 4. 6. The transfer completes. 7. (optional) The user queries the transfer service to obtain the transfer's performance information, including total transfer time, number of bytes and files transferred, and bandwidth utilization data.
Variations (optional)	[Note: The original variations (a) and (b) were treated elsewhere in this document.] c) Instead of specifying a list of files/directories to be transferred, the user may request that a source file or directory and a destination file or directory be synchronized, meaning that only files that differ between the source and destination are transferred.
Quality Attributes	a) Any request to the file transfer service is acknowledged within one second.[source: campus bridging] b) The file transfer service can support, XSEDE-wide, a request rate of up to 1 request/second in aggregate from all users without error. [Source: A&D] c) The file transfer service can support XSEDE-wide, without error, at least 1,000 active transfer requests (i.e., under management, could be pending, active, etc) requests. [Source: A&D] d) Once a file transfer completes, the user may check its status for at least one month. [Source: A&D.] e) Request patterns that exceed the stated request submission rate or total active request limits are handled gracefully: e.g., by declining to accept further requests.[Source: campus bridging] f) The file transfer service can be restarted (e.g., following failure or system administrator action) without losing track of file transfer requests that are queued or active at three Sigma.[source: campus bridging] g) Valid file transfer requests complete at two sigma. [source:campus bridging] h) The file transfer service will be available at three Sigma. [source: campus bridging] i) The file transfer service can restart a transfer that is interrupted by transient failures. [source: campus bridging] j) The combination of transfer efficiency and impact of failures and restarts provides efficiency that is at least as good as 50% of peak theoretically possible throughput of optimal network path and storage systems. [source: campus bridging] [The A&D team feels that this is a difficult to measure attribute, in particular, peak bandwidth of end-to-end system.] k) If a user's request can not be accommodated by the transfer service (for example, if the user attempts to access a file system for which access is not authorized or if there is insufficient storage on the destination), the error message returned should be consistent, meaningful, and helpful. [source:

	campus bridging] l) The transfer service should provide “at most once” semantics, meaning that if the user submits the same file transfer request more than once (e.g., because an earlier submission was not acknowledged), the transfer will be executed at most once. m) The transfer service must provide a user-selectable option for requesting that the destination preserves the modification and access timestamps and mode bits (assuming Unix on both ends) from the original file(s) n) The transfer service must provide a user-selectable option for enabling confidentiality in the transfer request. I.e., cryptographic methods are used to prevent anyone else from eavesdropping the data being transferred. o) The transfer service must provide data integrity checking on all transfers.
Non-functional (optional)	
Issues	