Authentication and Session Management

Table of Contents

1 API Specification 1
1.1 Requirements and Limitations 1
1.2 Create Session 1
1.3 Create Anonymous Session 2
1.4 Destroy Session 2

About this Document

Title:	Authentication and Session Management
Editor:	Bernd Simon (bernd.simon@wu-wien.ac.at) David Massart (david.massart@eun.org) Frans Van Assche (frans.van.assche@eun.org) Stefaan Ternier (stefaan.ternier@cs.kuleuven.ac.be) Erik Duval (erik.duval@cs.kuleuven.ac.be)
Status:	Public Draft
Version (Date):	1.0 Beta (2005-04-13)
Summary:	This specification presents an API for authentication and session management.
Document Location:	http://www.prolearn-project.org/lori

Revision History

Version	*Changes*
1.0	➢ Method signatures were modified to take the new exception model into account. ➢ The term ‘exception’ was replaced by the more general concept of ‘fault’. ➢ The java-like notation of the methods was replaced by a language neutral description.

Version	*Changes*
0.8	➢ The requirement that passwords need to be MD5 encoded was removed, since it was derived from a particular application setting. ➢ Session ID creation requirements were specified in more detail.

Acknowledgements

This work is partly sponsored by the CEN/ISSS Workshop on Learning Technologies. We also would like to acknowledge contributions from the following initiatives: Ariadne, Celebrate, Educanext, Edutella, Elena, EduSource, ProLearn, Universal, Zing.

List of Contributors

Bernd Simon, Vienna University of Economics and Business Administration & EducaNext
Christian Werner, Learning Lab Lower Saxony
Erik Duval, Katholieke Universiteit Leuven & Ariadne
Daniel Olmedilla, Learning Lab Lower Saxony
Dan Rehak, Carnegie Mellon University
David Massart, European Schoolnet
Frans Van Assche, European Schoolnet
Griff Richards, Simon Fraser University
Gerhard Müller, IMC
Julien Tane, Universität Karlsruhe
Marek Hatala, Simon Fraser University
Matthew J. Dovey, Oxford University
Peter Dolog, Learning Lab Lower Saxony
Stefaan Ternier, Katholieke Universiteit Leuven
Stefano Ceri, Politecnico Milano
Stefan Brantner, BearingPoint Infonova
Simos Retalis, University of Piraeus
Theo van Veen, Koninklijke Bibliotheek
Zoltán Miklós, Vienna University of Economics and Business Administration

Context

Currently, there are three SQI related documents:

i. "Learning Object Repository Interoperability Framework": this document provides the "big picture" and should probably be read first if you want to understand the context of this work;
ii. "Simple Query Interface Specification": this is the core of the specification for querying learning object repositories in an interoperable way;
iii. "Authentication and Session Management": this document focuses on specific issues related to authentication and session management.

1 API Specification

1.1 Requirements and Limitations

This specification introduces an incomplete list of possible means for establishing a session for the communication between two systems.

Once a session has been established it is assumed that the source has the right to communicate with the target. In order to establish a session, a user name and password or any other credential may be required. The identification of a source repository can prevent candidate target repositories from opening up their systems to unknown partners, and enables query policies.

A session is valid until it is destroyed. Hence, it continues to be active after a query has been executed. In order to destroy a session, the service `destroySession` must be called. Alternatively, a session times out when no communication takes place during 30 minutes. However, a session might be valid much longer than 30 minutes and sometimes might even require manual destruction.

The Session ID is decided in the application profile. One can imagine, for example, a randomly generated 128-bit value that is based on a random seed, which hashes several longs unique to the specific repository (e.g. IP, port, start time).

The specification assumes the use of secure authentication, authorization, and encryption mechanisms such as those provided by state-of-the-art technology (e.g., SSL).

1.2 Create Session

This method creates a session. It requires *userID* and *password* as parameters. The method returns a Session ID.

Method name	**createSession**	
Return type	String	
Parameters	**Name**	**Type**
	userID	String
	password	String
Fault	WRONG_CREDENTIALS METHOD_FAILURE	

An *SQIFault* is thrown with code:

- “WRONG_CREDENTIALS” in case an invalid *userID* and/or *password* was provided;
- “METHOD_FAILURE” in case the creation of the session failed due to any other reason (for example because of too many open sessions, etc.).

1.3 Create Anonymous Session

This method creates a session without requiring an account at the system where the session will be created. The method returns a Session ID.

Warning: The `createAnonymousSession` method has the potential to be used for denial-of-service attacks. Hence, implementers need to take appropriate measures, e.g. restricting the number of active anonymous sessions.

Method name	**createAnonymousSession**
Return type	String
Fault	METHOD_FAILURE

An *SQIFault* is thrown with code "SQI-00001" in case the creation of the session has failed.

1.4 Destroy Session

The system initiating a session can use this method in order to destroy a session. The parameter *sessionID* identifies the session.

Method name	**destroySession**	
Return type	Void	
Parameters	**Name**	**Type**
	sessionID	String
Fault	NO_SUCH_SESSION METHOD_FAILURE	

An *SQIFault* is thrown with code:

- "NO_SUCH_SESSION" when an invalid *sessionID* is supplied;
- "METHOD_FAILURE" in case the method failed due to any other reason.