OSID V3 Specifications
transaction package

Version Draft 3

This specifications represent a draft for OSID V3 interface definitions. These definitions may change at any time.

Last Modified: 4 July 2008

prepared by:
Tom Coppeto
OnTapSolutions

	OSID License
Copyright	Copyright © 2002-2008 Massachusetts Institute of Technology. All Rights Reserved.
License	This Work is being provided by the copyright holder(s) subject to the following license. By obtaining, using and/or copying this Work, you agree that you have read, understand, and will comply with the following terms and conditions. This Work and the information contained herein is provided on an "AS IS" basis. The Massachusetts Institute of Technology, the Open Knowledge Initiative, and THE AUTHORS DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE WORK OR THE USE OR OTHER DEALINGS IN THE WORK. Permission to use, copy and distribute unmodified versions of this Work, for any purpose, without fee or royalty is hereby granted, provided that you include the above copyright notice and the terms of this license on ALL copies of the Work or portions thereof. You may nodify or create Derivatives of this Work only for your internal purposes. You shall not distribute or transfer any such Derivative of this Work to any location or to any third party. For the purposes of this license, Derivative shall mean any derivative of the Work as defined in the United States Copyright Act of 1976, such as a translation or modification. The export of software employing encryption technology may require a specific license from the United States Government. It is the responsibility of any person or organization comtemplating export to obtain such a license before exporting this Work.

Package Description	osid.transaction package

osid.transaction.TransactionProfile
osid.transaction.TransactionManager
osid.transaction.TransactionProxyManager
osid.transaction.TransactionSession
osid.transaction.Transaction
osid.transaction.TransactionState

Package	**osid.transaction**
Title	**Transaction Open Service Interface Definitions**
Version	3.0.0

Description	The transaction OSID provides the means in which multiple OSID sessions can be coordinated. An OsidSession supports transactions by supporting the Transaction interface. The TransactionSession is the way multiple Transaction objects can be managed.
	
	An ideal transaction is one that supports Atomicity, Consistency, Isolation and Durability (ACID). This is difficult to achieve and impossible without the cooperation of the underlying system. The transaction interface, however, is a useful vehicle for supporting multiple operations or a sequence of operations. Since the ACID characteristics of a transaction, like all issues of data persistence, are in the purvue of the provider, they are not addressed in the interface definition leaving a given provider to determine the nature of the transactions it wants to support.
	
	An example:
	
	`Transaction agentTransaction = agentAdminSession.createTransaction();`
	`Transaction msgTransaction = messagingSession.createTransaction();`
	
	`TransactionSession transactions = manager.createSession();`
	`transactions.add(agentTransaction);`
	`transactions.add(msgTransaction);`
	
	`agentAdminSession.updateAgent(agentId, agentForm);`
	`messagingSession.send(agentId, "I changed your name.");`
	
	`transactions.commit();`

Interface	**osid.transaction.TransactionProfile**	
Implements	osid.OsidProfile	
Description	The TransactionProfile describes the interoperability among transaction services.	
Method	**supportsTransactions**	
Description	Tests if transactions are supported.	
Return	boolean	true if transactions are supported, false otherwise
Compliance	mandatory	This method must be implemented.

Interface	**osid.transaction.TransactionManager**	
Implements	osid.OsidManager	
	osid.transaction.TransactionProfile	
Description	Gets the OsidSession associated with the transaction service.	
Method	**getTransactionSession**	
Description	Gets the OsidSession associated with the transaction service.	
Return	osid.transaction.TransactionSession	a transaction session
Errors	OPERATION_FAILED	unable to complete request
	UNIMPLEMENTED	supportsTransactions() is false
Compliance	mandatory	This method must be implemented.

Interface	**osid.transaction.TransactionProxyManager**	
Implements	osid.OsidProxyManager	
	osid.transaction.TransactionProfile	
Description	Gets the OsidSession associated with the transaction service.	
Method	**getTransactionSession**	
Description	Gets the OsidSession associated with the transaction service.	
Parameters	osid.authentication.Authentication authentication	proxy authentication
Return	osid.transaction.TransactionSession	a transaction session
Errors	NULL_ARGUMENT	authentication is null
	OPERATION_FAILED	unable to complete request
	PERMISSION_DENIED	authentication is invalid
	UNIMPLEMENTED	supportsTransactions() is false
	UNSUPPORTED	authentication is not supported
Compliance	mandatory	This method must be implemented.

Interface	**osid.transaction.TransactionSession**		
Implements	osid.OsidSession		
Description	The transaction session is coordinate transactions. A transaction session allows for Transactions to be added to the list of transactions it is managing. Upon a commit(), all registered transactions receive a prepare() and a commit(). Upon an abort() all registered transactions receive an abort(). A TransactionSession itself may implement transactions (as it is an OsidSession) as a means of enabling a form of federated transaction management.		
Method	**add**		
Description	Adds a Transaction to be managed by this transaction service.		
Parameters	osid.transaction.Transaction	transaction	the transaction to add
Errors	ALREADY_EXISTS		transaction already added
	ILLEGAL_STATE		this transaction has ended
	INVALID_ARGUMENT		the session doesn't support transactions
	NULL_ARGUMENT		a null argument provided
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.
Method	**commit**		
Description	Commits the transaction and makes the state change(s) visible. This transaction is effectively closed and the only valid method that may be invoked is getState().		
Errors	ILLEGAL_STATE		this transaction has been committed or aborted
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.
Provider Notes	prepare() should be invoked on all regstered transactions and iff all are successful should commit() be invoked on all registered transactions. In case of error on any prepare(), all transactions should be aborted. If an error occurs on a commit() after a transaction reported success on a prepare() after one or more transactions were already committed, then it is not ACID compliant and success should be assumed by committing the rest of the transactions. If a commit() error occurs when no transactions have been committed, then this operation should not proceed.		
Method	**abort**		
Description	Cancels this transaction, rolling back the queue of operations since the start of this transaction. This transaction is effectively closed and the only valid method that may be invoked is getState().		
Errors	ILLEGAL_STATE		this transaction has been committed or aborted
Compliance	mandatory		This method must be implemented.
Provider Notes	Invokes abort() on all registered transactions().		
Method	**getState**		
Description	Gets the current state of this transaction.		
Return	osid.transaction.TransactionState		the current state of this transaction
Errors	ILLEGAL_STATE		this transaction has been committed or aborted
Compliance	mandatory		This method must be implemented.

Interface	**osid.transaction.Transaction**	
Implements		
Description	OsidTransaction is used by an OsidSession to provide transactions across its methods. Transactions are performed within a session. Coordination of transactions across OSIDS of there sessions requires the availability of a transaction manager. A trsnaction is started upon creation of an OsidTransaction. Actions within a session are queued until the transaction is committed or aborted.	
Method	**prepare**	
Description	Prepares for a commit. No further operations are permitted in the associated manager until this transaction is committed or aborted.	
Errors	ILLEGAL_STATE	this transaction has been committed or aborted
	OPERATION_FAILED	unable to complete request
	PERMISSION_DENIED	authorization failure
	TRANSACTION_FAILURE	this transaction cannot proceed due to a bad transaction element
Compliance	mandatory	This method must be implemented.
Provider Notes	The provider must verify this transaction such that a commit will succeed and reliably record the state changes resulting from this transaction before returning.	
Method	**commit**	
Description	Commits the transaction and makes the state change(s) visible. This transaction is effectively closed and the only valid method that may be invoked is getState().	
Errors	ILLEGAL_STATE	this transaction has been committed or aborted
	OPERATION_FAILED	unable to complete request
	PERMISSION_DENIED	authorization failure
Compliance	mandatory	This method must be implemented.
Provider Notes	Any resources allocated for this transaction can be released.	
Method	**abort**	
Description	Cancels this transaction, rolling back the queue of operations since the start of this transaction. This transaction is effectively closed and the only valid method that may be invoked is getState().	
Errors	ILLEGAL_STATE	this transaction has been committed or aborted
Compliance	mandatory	This method must be implemented.
Provider Notes	Any resources allocated for this transaction can be released.	
Method	**getState**	
Description	Gets the current state of this transaction.	
Return	osid.transaction.TransactionState	the current state of this transaction
Compliance	mandatory	This method must be implemented.

Enumeration	**osid.transaction.TransactionState**	
Description	This enumeration contains the possible state values of an OsidTransaction.	
Vaues	START	transaction has started and operations may be recorded
	COMMIT_READY	prepare() was successful and ready for a commit() or an abort()
	COMMITTED	the transaction committed successfully
	ABORTED	the transaction has been aborted. The implementation may set an ABORTED state if an error in processing the transaction has occurred.