OSID V3 Specifications
osid package

Version Draft 3

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

Last Modified: 4 October 2008

prepared by:
Tom Coppeto
OnTapSolutions

	OSID License
Copyright	Copyright © 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 package

Interfaces	osid.OsidProfile osid.OsidManager osid.OsidProxyManager osid.OsidSession osid.OsidObject osid.OsidQuery osid.OsidForm osid.OsidSearchOrder osid.OsidRecord osid.OsidSearch osid.OsidSearchResults osid.OsidSearchRecord osid.OsidReceiver osid.OsidList osid.OsidCatalog osid.OsidCatalogQuery osid.OsidCatalogForm osid.OsidCatalogSearchOrder osid.OsidRuntimeProfile osid.OsidRuntimeManager osid.Metadata osid.Property osid.PropertyList osid.SpatialUnit osid.SpatialUnitList osid.ServiceReceiver

Enumerations	osid.OSID osid.MetadataSyntax

Package	**osid**
Title	**Core Service Interface Definitions**
Version	3.0.0
Description	The osid package defines how an application loads a service and includes common definitions used throughout the OSIDs. The osid package consists of: • OsidProfile: The OsidProfile defines the interoperability tests for an OSID. • OsidRuntimeManager: The OsidRuntimeManager defines an interface to instantiate and initialize an instance of an OSID implementation. • OsidManager: The OsidManager defines an interface for methods in common throuhgout the various OSID managers. An OSID manager is the principal control point that profiles supported services and types, and is responsible for session creation. OSID managers are created through the OsidRuntimeEnvironment. • OsidProxyManager: A variant of an OsidManager for methods that support proxy authentication objects. • OsidSession: The OsidSession defines an interface for methods in common throughout the various OSID sessions. An OSID session contains method definitions for an aspect of a service. OSID sessions are created through OSID managers. • OsidObject: The OsidObject defines an interface for methods in common throughout the various OSID objects. An OSID object defines a set of object data. OSID objects are accessed from OSID sessions. • OsidQuery: The OsidQuery defines an interface in common throughout the various OSID queries. An OsidQuery defines a set of methods to query an OSID for its OsidObjects. • OsidSearch: The OsidSearch defines an interface in common throughout vrious OSID searches. An OsidSearch defines a set of methods to manage search options for performing searches. • OsidSearch: The OsidSearch defines an interface in common throughout vrious OSID searches. OsidSearchResults defines a set of methods to manage search results. • OsidForm: The OsidForm defines an interface for methods in common throughout the various OSID forms. An OSID form defines a set of methods to modify data in an OSID object. OSID forms are accessed from OSID sessions. • OsidReceiver: The OsidReceiver defines an interface in common throughout the OSID receivers. An OsidReceiver defines a set of methods invoked for asynchronous notification. • OsidList: The OsidList defines an interface for methods in common throughout the various lists. An OsidList defines a set of methods to sequentially access a set of objects. OSID lists are accessed from OSID sessions. • OSID: enumerates the list of supported OSIDs • Metadata: defines a set of methods for describing a data element to provide application hints for the creation and updating of the data element • Primitive: enumerates the list of supported primitive types for describing metadata • TimeResolution: enumerates a list of tiem resolutions for use with metadata in date/time data elements • Property: Maps a name to a value. Properties are available in OSID objects to provide a simplified view of data that may exist within a typed interface. • PropertyList: A list of properties.

	Generally, these definitions are not accesed directly but are used to define interfaces in the OSIDs themselves. OSIDs derive most of their definitions from a definition in the osid package. What methods appear in the interfaces at this level versus an actual OSID is determined by the typing in the method signatures. The osid package interfaces are a means of ensuring consistency of common methods and not designed to facilitate object polymorphism among different OSIDs. A language binder may elect to alter the interface hierarchy presented in this specification and a provider need not parallel these interfaces in their implementations. The flow of control through any OSID can be described in terms of these definitions. An OsidManager or OsidProxyManager is retrieved from the OsidRuntimeManager for a given service. Both types of managers share an interface for describing what they support in the OsidProfile. OsidSessions are created from the OsidManager. OsidSessions tend to be organized along clusters of like-functionality. Lookup-oriented sessions retrieve OsidObjects. Return of multiple OsidObjects is done via the OsidList. Search-oriented sessions retrieve OsidObjects through searches provided through the OsidQuery and OsidSearch interfaces. Administrative-oriented sessions create and update OsidObjects using the OsidForm interface. The OsidForm makes available Metadata to help define its rules for setting and changing various data elements. A notification session provides a means for subscribing to events, "a new object has been created", for example, and these events are received from an OsidReceiver.

Interface	**osid.OsidProfile**	
Implements		
Description	The OsidProfile defines the interoperability areas of an OSID. An OsidProfile is implemented by an OsidManager. The top level OsidProfile tests for version compatibility. Each OSID extends this interface to include its own interoperability definitions within its managers.	
Method	**getId**	
Description	Gets an identifier for this service implementation. The identifier is unique among services but multiple instantiations of the same service use the same Id. This identifier is the same identifier used in managing OSID installations.	
Return	osid.id.Id	the Id
Compliance	mandatory	This method must be implemented.
Method	**getDisplayName**	
Description	Gets a display name for this service implementation.	
Return	string	a display name
Compliance	mandatory	This method must be implemented.
Method	**getDescription**	
Description	Gets a description of this service implementation.	
Return	string	a description
Compliance	mandatory	This method must be implemented.
Method	**getVersion**	
Description	Gets the version of this service implementation.	
Return	string	the version
Compliance	mandatory	This method must be implemented.
Method	**getReleaseDate**	
Description	Gets the date this service implementation was released.	
Return	osid.calendaring.DateTime	the release date
Compliance	mandatory	This method must be implemented.
Method	**getLicense**	
Description	Gets the terms of usage with respect to this service implementation.	
Return	string	the license
Compliance	mandatory	This method must be implemented.
Method	**getProviderId**	
Description	Gets the Resource Id representing the provider of this service.	
Return	osid.id.Id	the provider Id
Compliance	mandatory	This method must be implemented.
Method	**getProvider**	
Description	Gets the provider of this service, expressed using the Resource interface.	
Return	osid.resource.Resource	the service provider resource
Errors	OPERATION_FAILED	unable to complete request
Compliance	mandatory	This method must be implemented.
Provider Notes	The Resource at minimum may only contain some identifier along with a name and description, or a typed interface extension can be used to reveal more information such as contact information about the provider.	
Method	**getBranding**	
Description	Gets a branding, such as an image or logo, expressed using the Asset interface.	
Return	osid.repository.AssetList	a list of assets
Errors	OPERATION_FAILED	unable to complete request
Compliance	mandatory	This method must be implemented.

Method	supportsOSIDVersion		
Description	Test for support of an OSID version.		
Parameters	string	version	the version string to test
Return	boolean		true if this manager supports the given version, false otherwise
Compliance	mandatory		This method must be implemented.
Provider Notes	An implementation may support multiple versions of an OSID.		

Method	supportsJournaling	
Description	Test for support of a journaling service.	
Return	boolean	true if this manager supports the journaling, false otherwise
Compliance	mandatory	This method must be implemented.

Interface	**osid.OsidManager**		
Implements	osid.OsidProfile		
Description	The OsidManager is the top level interface for all OSID managers. An OSID manager is instantiated through the OsidRuntimeManager and represents an instance of a service. An OSID manager is responsible for implementing a profile for a service and creating sessions that, in general, correspond to the profile. An application need only create a single OsidManager per service and implementors must ensure the OsidManager is thread-safe. The OsidSessions spawned from an OSID manager are dedicated to single processing threads. The OsidManager defines methods in common throughout all OSID managers which implement this interface.		
Method	**initialize**		
Description	Initializes this manager. A manager is initialized once at the time of creation.		
Parameters	osid.OsidRuntimeManager	runtime	the runtime environment
Errors	CONFIGURATION_ERROR		an error with implementation configuration
	ILLEGAL_STATE		this manager has already been initialized by the OsidLoader
	NULL_ARGUMENT		runtime is null
	OPERATION_FAILED		unable to complete request
Compliance	mandatory		This method must be implemented.
Provider Notes	In addition to loading its runtime configuration an implementation may create shared resources such as connection pools to be shared among all sessions of this service and released when this manager is closed. Providers must thread-protect any data stored in the manager. To maximize interoperability, providers should not honor a second call to initialize() and must set an ILLEGAL_STATE error.		
Method	**getJournalSession**		
Description	Gets the Journal session for this service.		
Return	osid.journaling.JournalSession		a journal session
Errors	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure occurred
	UNIMPLEMENTED		supportsJournaling() is false
Compliance	mandatory		This method must be implemented.
Method	**rollbackService**		
Description	Rolls back this service to a point in time.		
Parameters	timestamp	rollbackTime	the requested time
Return	osid.journaling.JournalEntry		the journal entry corresponding to the actual state of this service
Errors	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure occurred
	UNIMPLEMENTED		supportsJournaling() is false
Compliance	mandatory		This method must be implemented.
Method	**getServiceMessage**		
Description	Gets a service message which can be used for service announcements.		
Return	string		service message
Errors	OPERATION_FAILED		unable to complete request
Compliance	mandatory		This method must be implemented.

Method	registerForServiceMessages		
Description	Register for service messages. ServiceMessage.newMessage() is invoked for each new message. There is a single service message receiver per manager.		
Parameters	osid.ServiceReceiver	receiver	supplied interface for service messages
Errors	NULL_ARGUMENT		receiver is null
	OPERATION_FAILED		unable to complete request
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidProxyManager**		
Implements	osid.OsidProfile		
Description	The OsidProxyManager is the top level interface for all OSID proxy authentication managers. A proxy manager accepts parameters to pass through end-user authentication credentials if necessary in a server environment. This pass-through inherently couples a provider and consumer together by way of the authentication technology. Native applications should use an OsidManager to maintain a higher degree of interoperability by avoiding this coupling. An OSID proxy manager is instantiated through the OsidRuntimeManager and represents an instance of a service. An OSID manager is responsible for defining clusters of interoperability within a service and creating sessions that generally correspond to these clusters, An application need only create a single OsidProxyManager per service and implementors must ensure the OsidProxyManager is thread-safe. The OsidSessions spawned from an OSID manager are dedicated to single processing threads. The OsidProxyManager defines methods in common throughout all OSID managers which implement this interface.		
Method	**initialize**		
Description	Initializes this manager. A manager is initialized once at the time of creation.		
Parameters	osid.OsidRuntimeManager	runtime	the runtime environment
Errors	CONFIGURATION_ERROR		an error with implementation configuration
	ILLEGAL_STATE		this manager has already been initialized by the OsidLoader
	NULL_ARGUMENT		runtime is null
	OPERATION_FAILED		unable to complete request
Compliance	mandatory		This method must be implemented.
Provider Notes	In addition to loading its runtime configuration an implementation may create shared resources such as connection pools to be shared among all sessions of this service and released when this manager is closed. Providers must thread-protect any data stored in the manager. To maximize interoperability, providers should not honor a second call to initialize() and must set an ILLEGAL_STATE error.		
Method	**getJournalSession**		
Description	Gets the Journal session for this service.		
Parameters	osid.authentication.Authentication	authentication	a proxy authentication
Return	osid.journaling.JournalSession		a journal session
Errors	NULL_ARGUMENT		authentication is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure occurred
	UNIMPLEMENTED		supportsJournaling() is false
	UNSUPPORTED		authentication is not supported
Compliance	mandatory		This method must be implemented.

Method	rollbackService		
Description	Rolls back this service to a point in time.		
Parameters	timestamp	rollbackTime	the requested time
	osid.authentication.Authentication	authentication	a proxy authentication
Return	osid.journaling.JournalEntry		the journal entry corresponding to the actual state of this service
Errors	NULL_ARGUMENT		authentication is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure occurred
	UNIMPLEMENTED		supportsJournaling() is false
	UNSUPPORTED		authentication is not supported
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidSession**	
Implements		
Description	The OsidSession is the top level interface for all OSID sessions. An OsidSession is created through its corresponding OsidManager. A new OsidSession should be created for each user of a service and for each processing thread. A session maintains a single authenticated user and is not required to ensure thread-protection. A typical OSID session defines a set of service methods corresponding to some compliance level as defined by the service and is generally responsible for the management and retrieval of OsidObjects. OsidSession defines a set of common methods used throughout all OSID sessions. An OSID session may optionally support transactions through the transaction interface.	
Method	**isAuthenticated**	
Description	Tests if there are valid authentication credentials used by this service.	
Return	boolean	true if valid authentication credentials exist, false otherwise
Compliance	mandatory	This method must be implemented.
Provider Notes	Providers must also query OsidSessions instantiated by this session.	
Method	**getAuthenticatedAgents**	
Description	Gets the authenticated identities used by this service to give the user feedback as to which of the Agent identitites are actively being used on the user's behalf.	
Return	osid.authentication.AgentList	the list of authenticated Agents
Compliance	mandatory	This method must be implemented.
Provider Notes	Providers must also include any authenticated Agents from all OsidSessions instantiated by this service.	
Method	**supportsTransactions**	
Description	Tests for the availability of transactions.	
Return	boolean	true if transaction methods are available, false otherwise
Compliance	mandatory	This method must be implemented.
Method	**startTransaction**	
Description	Starts a new transaction for this sesson. Transactions are a means for an OSID to provide an all-or-nothing set of operations within a session and may be used to coordinate this service from an external transaction manager. A session supports one transaction at a time. Starting a second transaction before the previous has been committed or aborted results in an ILLEGAL_STATE error.	
Return	osid.transaction.Transaction	a new transaction
Errors	ILLEGAL_STATE	a transaction is already open
	OPERATION_FAILED	unable to complete request
	UNSUPPORTED	transactions not supported
Compliance	optional	This method must be implemented if supportsTransactions() is true.
Provider Notes	Ideally, a provider that supports transactions should guarantee atomicity, consistency, isolation and durability in a 2 phase commit process. This is not always possible in distributed systems and a transaction provider may simply allow for a means of processing bulk updates. To maximize interoperability, providers should honor the one-transaction-at-a-time rule.	

Interface	**osid.OsidObject**
Implements	
Description	OsidObject is the top level interface for all OSID objects. An OSID object is an object identified by an OSID Id and may implements optional interfaces. OSID objects also contain a display name and a description. These fields are required but may be used for a variety of purposes ranging from a primary name and description of the object to a more user friendly display of various attributes. Creation of OSID objects and the modification of their data is managed through the associated OsidSession which removes the dependency of updating data elements upon object retrieval. The OsidManager should be used to test if updates are available and determine what PropertyTypes are supported. The OsidManager is also used to create the appropriate OsidSession for object creation, updates and deletes. All OsidObjects are identified by an immutable Id. An Id is assigned to an object upon creation of the object and cannot be changed once assigned. An OSID object may support one or more supplementary records which are expressed in the form of interfaces. Each record interface is identified by a Type. A record interface may extend another record interface where support of the parent record interface is implied. In this case of interface inheritance, support of the parent record type may be implied through hasRecordType() and not explicit in getRecordTypes(). For example, if recordB extends recordA, typeB is a child of typeA. If a record implements typeB, than it also implements typeA. An application that only knows about typeA retrieves recordA. An application that knows about typeB, retrieves recordB which is the union of methods specified in typeA and typeB. If an application requests typeA, it may not attempt to access methods defined in typeB as they may not exist until explicitly requested. The mechanics of this polymorphism is defined by the language binder. One mechanism might be the use of casting. In addition to the record Types, OSID objects also have a genus Type. A genus Type indicates a classification or kind of the object where an "is a" relationship exists. The purpose of of the genus Type is to avoid the creation of unnecessary record types that may needlessly complicate an interface hierarchy or introduce interoperability issues. For example, an OSID object may have a record Type of Publication that defines methods pertinent to publications, such as an ISBN number. A provider may wish to distinguish between books and journals without having the need of new record interfaces. In this case, the genus Type may be one of Book or Journal. While this distinction can aid a search, these genres should be treated in such a way that do not introduce interoperability problems. Like record Types, the genus Types may also exist in an implicit type hierarchy. An OSID object always has at least one genus. Genus types should not be confused with subject tagging, which is managed externally to the object. Unlike record Types, an object's genus may be modified. However, once an object's record is created with a record Type, it cannot be changed. Methods that return values are not permitted to return nulls. If a value is not set, it is indicated in the Metadata of the update form.

Method	getId	
Description	Gets the Id associated with this instance of this OSID object. Persisting any reference to this object is done by persisting the Id returned from this method. The Id returned may be different than the Id used to query this object. In this case, the new Id should be preferred over the old one for future queries.	
Return	osid.id.Id	the Id
Compliance	mandatory	This method must be implemented.
Provider Notes	The Id is intended to be constant and persistent. A consumer may at any time persist the Id for retrieval at any future time. Ideally, the Id should consistently resolve into the designated object and not be reused. In cases where objects are deactivated after a certain lifetime the provider should endeavor not to obliterate the object or its Id but instead should update the properties of the object including the deactiavted status and the elimination of any unwanted pieces of data. As such, there is no means for updating an Id and providers should consider carefully the identification scheme to implement. Id assignments for objects are strictly in the realm of the provider and any errors should be fixed directly with the backend supporting system. Once an Id has been assigned in a production service it should be honored such that it may be necessary for the backend system to support Id aliasing to redirect the lookup to the current Id. Use of an Id OSID may be helpful to accomplish this task in a modular manner.	

Method	getDisplayName	
Description	Gets the preferred display name associated with this instance of this OSID object appropriate for display to the user.	
Return	string	the display name
Compliance	mandatory	This method must be implemented.
Provider Notes	A display name is a string used for identifying an object in human terms. A provider may wish to initialize the display name based on one or more object attributes. In some cases, the display name may not map to a specific or significant object attribute but simply be used as a preferred display name that can be modified. A provider may also wish to translate the display name into a specific locale using the Locale service. Some OSIDs define methods for more detailed naming.	

Method	getDescription	
Description	Gets the description associated with this instance of this OSID object.	
Return	string	the description
Compliance	mandatory	This method must be implemented.
Provider Notes	A description is a string used for describing an object in human terms and may not have significance in the underlying system. A provider may wish to initialize the description based on one or more object attributes and/or treat it as an auxiliary piece of data that can be modified. A provider may also wish to translate the description into a specific locale using the Locale service.	

Method	getRecordTypes	
Description	Gets the record types available in this object. A record Type explicitly indicates the specification of an interface to the record. A record may or may not inherit other record interfaces through interface inheritance in which case support of a record type may not be explicit in the returned list. Interoperability with the typed interface to this object should be performed through hasRecordType().	
Return	osid.type.TypeList	the record types available through this object
Compliance	mandatory	This method must be implemented.

Method	hasRecordType		
Description	Tests if this object supports the given record Type. The given record type may be supported by the object through interface/type inheritence. This method should be checked before retrieving the record interface.		
Parameters	osid.type.Type	recordType	a type
Return	boolean		true if a record of the given record Type is available, false otherwise
Errors	NULL_ARGUMENT		recordType is null
Compliance	mandatory		This method must be implemented.

Method	getGenusType	
Description	Gets the genus type of this object.	
Return	osid.type.Type	the genus type of this object
Compliance	mandatory	This method must be implemented.

Method	isOfGenusType		
Description	Tests if this object is of the given genus Type. The given genus type may be supported by the object through the type hierarchy.		
Parameters	osid.type.Type	genusType	a genus type
Return	boolean		true if this object is of the given genus Type, false otherwise
Errors	NULL_ARGUMENT		genusType is null
Compliance	mandatory		This method must be implemented.

	isCurrent	
Description	Tests to see if the last method invoked retrieved up-to-date data. Simple retrieval methods do not specify errors as, generally, the data is retrieved once at the time this object is instantiated. Some implementations may provide real-time data though the application may not always care. An implementation providing a real-time service may fall back to a previous snapshot in case of error. This method returns false if the data last retrieved was stale.	
Return	boolean	true if the last data retrieval was up to date, false otherwise
Compliance	mandatory	This method must be implemented.
Provider Notes	Providers should return false unless all getters are implemented using real-time queries, or some trigger process keeps the data in this object current. Providers should populate basic data elements at the time this object is instantiated, or set an error, to ensure some data availability.	

Method	getProperties	
Description	Gets a list of all properties of this object including those corresponding to data within this object's records. Properties provide a means for applications to display a representation of the contents of an object without understanding its record interface specifications. Applications needing to examine a specific property or perform updates should use the methods defined by the object's record Type.	
Return	osid.PropertyList	a list of properties
Errors	OPERATION_FAILED	unable to complete request
	PERMISSION_DENIED	an authorization failure occurred
Compliance	mandatory	This method must be implemented.

Method	getPropertiesByRecordType		
Description	Gets a list of properties corresponding to the specified record type. Properties provide a means for applications to display a representation of the contents of an object without understanding its record interface specifications. Applications needing to examine a specific property or perform updates should use the methods defined by the object record Type. The resulting set includes properties specified by parents of the record type in the case a record's interface extends another.		
Parameters	osid.type.Type	recordType	the record type corresponding to the properties set to retrieve
Return	osid.PropertyList		a list of properties
Errors	NULL_ARGUMENT		recordType is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		an authorization failure occurred
	UNSUPPORTED		hasRecordType(recordType) is false
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidQuery**
Implements	

Description	The OsidQuery is used to assemble search queries. An OsidQuery is available from an OsidSession and defines methods to query for an OsidObject that includes setting a display name and a description. Once the desired parameters are set, the OsidQuery is given to the designated search method. The same OsidQuery returned from the session must be used in the search as the provider may utilize implementation-specific data wiithin the object.
	
	If multiple data elements are set in this interface, the results matching all the given data (eg: AND) are returned. Search methods throughout the OSIDs accept multiple OsidQuery interfaces. Each OsidQuery in the array behaves like an OR such that results are returned that match any of the given OsidQuery objects.
	
	Any match method inside an OsidQuery may be invoked multiple times. In the case of a match method, each invocation adds an element to an OR expression. Any of these terms may also be negated through the match flag.
	
	`OsidQuery { OsidQuery.matchDisplayName AND (OsidQuery.matchDescription OR OsidQuery.matchDescription)} OR OsidQuery`
	
	OsidObjects allow for the definition of an additonal records and the OsidQuery parallels this mechanism. An interface type of an OsidObject record must also define the corresponding OsidQuery record which is available through query interfaces. Multiple requests of these typed interfaces may return the same underlying object and thus it is only useful to request once.
	
	String searches are described using a string search Type that indicates the type of regular expression or wildcarding encoding. Compatibility with a strings search Type can be tested within this interface.
	
	As with all aspects of OSIDs, nulls cannot be used. Separate tests are available for querying for unset values except for required fields.
	
	An example to find all objects whose name starts with "Fred" plus objects whose name starts with "Barney", color is "purple" and the word "dinosaur" does not appear in the description. ColorQuery is an interface extension of the object that defines a color.
	
	`ObjectQuery queries[2];`
	`queries[0] = session.getObjectQueryForInterfaceType(colorType);`
	`queries[0].addDisplayNameMatch("Fred*", wildcardStringMatchType, true);`
	
	`queries[1] = session.getObjectQuery();`
	`queries[1].addDisplayNameMatch("Barney*", wildcardStringMatchType, true);`
	`queries[1].addDescriptionMatch("dinosaur", wordStringMatchType, false);`
	
	`ColorQuery recordQuery;`
	`recordQuery = queries[1].getObjectRecord(colorRecordType);`
	`recordQuery.matchColor("purple");`
	`ObjectList list = session.getObjectsByQuery(queries);`

Method	getStringMatchTypes		
Description	Gets the string matching types supported. A string match type specifies the syntax of the string query, such as matching a word or including a wildcard or regular expression.		
Return	osid.type.TypeList		a list containing the supported string match types
Compliance	mandatory		This method must be implemented.

Method	supportsStringMatchType		
Description	Tests if the given string matching type is supported.		
Parameters	osid.type.Type	searchType	a Type indicating a string match type
Return	boolean		true if the given Type is supported, false otherwise
Compliance	mandatory		This method must be implemented.

Method	matchKeyword		
Description	Adds a keyword to match. Multiple keywords can be added to perform a boolean OR among them. A keyword may be applied to any of the elements defined in this object such as the display name, description or any method defined in an interface implemented by this object.		
Parameters	string	keyword	keyword to match
	osid.type.Type	stringMatchType	the string match type
	boolean	match	true for a positive match, false for a negative match
Errors	INVALID_ARGUMENT		keyword is not of stringMatchType
	NULL_ARGUMENT		keyword or stringMatchType is null
	UNSUPPORTED		supportsStringMatchType(stringMatchType) is false
Compliance	mandatory		This method must be implemented.

Method	matchDisplayName		
Description	Adds a display name to match. Multiple display name matches can be added to perform a boolean OR among them.		
Parameters	string	displayName	display name to match
	osid.type.Type	stringMatchType	the string match type
	boolean	match	true for a positive match, false for a negative match
Errors	INVALID_ARGUMENT		keyword is not of stringMatchType
	NULL_ARGUMENT		displayName or stringMatchType is null
	UNSUPPORTED		supportsStringMatchType(stringMatchType) is false
Compliance	mandatory		This method must be implemented.

Method	matchDescription		
Description	Adds a description name to match. Multiple description matches can be added to perform a boolean OR among them.		
Parameters	string	description	description to match
	osid.type.Type	stringMatchType	the string match type
	boolean	match	true for a positive match, false for a negative match
Errors	INVALID_ARGUMENT		keyword is not of stringMatchType
	NULL_ARGUMENT		description or stringMatchType is null
	UNSUPPORTED		supportsStringMatchType(stringMatchType) is false
Compliance	mandatory		This method must be implemented.

Method	matchUnsetDescription		
Description	Matches a description that has no value.		
Compliance	mandatory		This method must be implemented.

Method	matchGenusType		
Description	Sets a Type for querying objects of a given genus. A genus type matches if the specified type is the same genus as the object genus type.		
Parameters	osid.type.Type	genusType	the object genus type
	boolean	match	true for a positive match, false for a negative match
Errors	NULL_ARGUMENT		genusType is null
Compliance	mandatory		This method must be implemented.

Method	matchParentGenusType		
Description	Sets a Type for querying objects of a given genus. A genus type matches if the specified type is the same genus as the object or if the specified type is an ancestor of the object genus in a type hierarchy.		
Parameters	osid.type.Type	genusType	the object genus type
	boolean	match	true for a positive match, false for a negative match
Errors	NULL_ARGUMENT		genusType is null
Compliance	mandatory		This method must be implemented.

Method	matchUnsetGenusType	
Description	Matches a genus that has no value.	
Compliance	mandatory	This method must be implemented.

Method	matchRecordType		
Description	Sets a Type for querying objects having records implementing a given record type. This includes records of the same interface type as the one provided and records implementing an ancestor interface type in an interface hierarchy.		
Parameters	osid.type.Type	recordType	the record interface type
	boolean	match	true for a positive match, false for a negative match
Errors	NULL_ARGUMENT		recordType is null
Compliance	mandatory		This method must be implemented.

Method	hasRecordType		
Description	Tests if this query supports the given record Type. The given record type may be supported by the object through interface/type inheritence. This method should be checked before retrieving the record interface.		
Parameters	osid.type.Type	recordType	a type
Return	boolean		true if a record query of the given record Type is available, false otherwise
Errors	NULL_ARGUMENT		recordType is null
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidForm**	
Implements		

Description	The OsidForm is used to create and update OsidObjects. The form is not an OsidObject but merely a container for data to be sent to an update or create method of a session. A provider may or may not combine the OsidObject and OsidForm interfaces into a single object.
	
	Generally, a set method parallels each get method of an OsidObject. Additionally, Metadata may be examined for each data element to assist in understanding particular rules concerning acceptable data.
	
	The form may provide some feedback as to the validity of certain data updates before the update transaction is issued to the correspodning session but a successful modification of the form is not a guarantee of success for the update transaction. A consumer may elect to perform all updates within a single update transaction or break up a large update intio smaller units. The tradeoff is the granularity of error feedback vs. the performance gain of a single transaction.
	
	As with all aspects of the OSIDs, nulls cannot be used. Methods to clear values are also defined in the form.
	A new OsidForm should be acquired for each transaction upon an OsidObject. Forms should not be reused from one object to another even if the supplied data is the same as the forms may encapsulate data specific to the object requested. Example of changing a display name and a color defined in a color interface extension:
	
	`ObjectForm form = session.getObjectFormForUpdate(objectId);`
	`form.setDisplayName("new name");`
	`ColorForm recordForm = form.getFormRecord(colorRecordType);`
	`recordForm.setColor("green");`
	`session.updateObject(objectId, form);`	

Method	**getCommentMetadata**	
Description	Gets the metadata for the comment corresponding to this form submission. The comment is used for describing the nature of the change to the corresponding object for the purposes of logging and auditing.	
Return	osid.Metadata	metadata for the comment
Compliance	mandatory	This method must be implemented.
Method	**setComment**	
Description	Sets a comment.	
Parameters	string comment	the new comment
Errors	INVALID_ARGUMENT	comment is invalid
	NO_ACCESS	comment cannot be modified
	NULL_ARGUMENT	comment is null
Compliance	mandatory	This method must be implemented.
Method	**isValid**	
Description	Tests if ths form is in a valid state for submission. A form is valid if all required data has been supplied compliant with any constraints.	
Return	boolean	false if there is a known error in this form, true otherwise
Errors	OPERATION_FAILED	attempt to perform validation failed
Compliance	mandatory	This method must be implemented.
Method	**getValidationMessage**	
Description	Gets a text message corresponding to additional instructions to pass form validation.	
Return	string	message
Compliance	mandatory	This method must be implemented.

Method	getDisplayNameMetadata		
Description	Gets the metadata for a display name.		
Return	osid.Metadata		metadata for the display name
Compliance	mandatory		This method must be implemented.
Method	setDisplayName		
Description	Sets a display name. A display name is required and if not set, will be set by the provider.		
Parameters	string	displayName	the new display name
Errors	INVALID_ARGUMENT		displayName is invalid
	NO_ACCESS		displayName cannot be modified
	NULL_ARGUMENT		displayName is null
Compliance	mandatory		This method must be implemented.
Method	getDescriptionMetadata		
Description	Gets the metadata for a description.		
Return	osid.Metadata		metadata for the description
Compliance	mandatory		This method must be implemented.
Method	setDescription		
Description	Sets a description.		
Parameters	string	description	the new description
Errors	INVALID_ARGUMENT		description is invalid
	NO_ACCESS		description cannot be modified
	NULL_ARGUMENT		description is null
Compliance	mandatory		This method must be implemented.
Method	clearDescription		
Description	Clears the description.		
Errors	NO_ACCESS		description cannot be modified
Compliance	mandatory		This method must be implemented.
Method	getGenusMetadata		
Description	Gets the metadata forr a genus type.		
Return	osid.Metadata		metadata for the genus
Compliance	mandatory		This method must be implemented.
Method	setGenusType		
Description	Sets a genus. A genus cannot be cleared because all objects have at minimum a root genus.		
Parameters	osid.type.Type	genusType	the new genus
Errors	INVALID_ARGUMENT		genusType is invalid
	NO_ACCESS		genusType cannot be modified
	NULL_ARGUMENT		genusType is null
Compliance	mandatory		This method must be implemented.
Method	hasRecordType		
Description	Tests if this form supports the given record Type. The given record type may be supported by the object through interface/type inheritence. This method should be checked before retrieving the record interface.		
Parameters	osid.type.Type	recordType	a record type
Return	boolean		true if a record form of the given record Type is available, false otherwise
Errors	NULL_ARGUMENT		recordType is null
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidSearchOrder**		
Implements			

Description	OsidSearchOrder specifies preferred ordering of search results.. An OsidSearchOrder is available from an search session and supplied to an OsidSearch interface.
	
	`OsidSearch os = session.getObjectSearch();`
	`os.limitResultSet(1, 25);`
	
	`OsidSearchOrder order = session.getObjectSearchOrder();`
	`order.orderByDisplayName();`
	`os.orderResults(order);`
	
	`OsidQuery queries[1];`
	`queries[0] = session.getObjectQuery();`
	`queries[0].addDescriptionMatch("*food*", wildcardStringMatchType, true);`
	
	`ObjectSearchResults results = session.getObjectsBySearch(queries, os);`
	`ObjectList list = results.getObjectList();`		

Method	**ascend**		
Description	Specifies a preference for ordering the result set in an ascending manner.		
Compliance	mandatory		This method must be implemented.
Method	**descend**		
Description	Specifies a preference for ordering the result set in a descending manner.		
Compliance	mandatory		This method must be implemented.
Method	**orderByDisplayName**		
Description	Specifies a preference for ordering the result set by the display name.		
Compliance	mandatory		This method must be implemented.
Method	**orderByGenusType**		
Description	Specifies a preference for ordering the result set by the genus type.		
Compliance	mandatory		This method must be implemented.
Method	**hasRecordType**		
Description	Tests if this search order supports the given record Type. The given record type may be supported by the object through interface/type inheritence. This method should be checked before retrieving the record interface.		
Parameters	osid.type.Type	recordType	a type
Return	boolean		true if an order record of the given record Type is available, false otherwise
Errors	NULL_ARGUMENT		recordType is null
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidRecord**		
Implements			
Description	OsidRecord is a top-level interface for all record objects. A record is an auxiliary interface that can be retrieved from an OSID object, query, form or search odrer that contains method definitions outside the core OSID specification. An OSID record interface specification is identified with a Type.		
Method	**implementsRecordType**		
Description	Tests if the given type is implemented by this record. Other types than that directly indicated by getType() may be supported through an inheritance scheme where the given type specifies a record that is a parent interface of the interface specified by getType().		
Parameters	osid.type.Type	recordType	a type
Return	boolean		true if the given record Type is implemented by this record, false otherwise
Errors	NULL_ARGUMENT		recordType is null
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidSearch**		
Implements			

Description	OsidSearch specifies search options used to perform OSID searches. An OsidSearch is available from an OsidSession and defines methods to govern the overall search of terms supplied in one or more OsidQuery interfaces.
	
	This interface is available from a search session.Example us using the search interface to retrieve the first 25 results:
	
	`OsidSearch os = session.getObjectSearch();`
	`os.limitResultSet(1, 25);`
	
	`OsidQuery queries[1];`
	`queries[0] = session.getObjectQuery();`
	`queries[0].addDescriptionMatch("*food*", wildcardStringMatchType, true);`
	
	`ObjectSearchResults results = session.getObjectsBySearch(queries, os);`
	`ObjectList list = results.getObjectList();`		

Method	**limitResultSet**		
Description	By default, searches return all matching results. This method restricts the number of results by setting the start and end of the result set, starting from 1. The starting and ending results can be used for paging results when a certain ordering is requested. The ending position must be greater than the starting position.		
Parameters	cardinal	start	the start of the result set
	cardinal	end	the end of the result set
Errors	INVALID_ARGUMENT		end is less than or equal to start
Compliance	mandatory		This method must be implemented.
Method	**hasSearchRecordType**		
Description	Tests if this search supports the given record Type. The given record type may be supported by the object through interface/type inheritence. This method should be checked before retrieving the record interface.		
Parameters	osid.type.Type	searchRecordType	a type
Return	boolean		true if a search record the given record Type is available, false otherwise
Errors	NULL_ARGUMENT		searchRecordType is null
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidSearchResults**		
Implements			

Description	This interface provides a means to capture results of a search and is used as a vehicle to perform a search within a previous result set. An example of searching withina result set:
	
	`OsidSearch os = session.getObjectSearch();`
	
	`OsidQuery queries[1];`
	`queries[0] = session.getAgentQuery();`
	`queries[0].addDescriptionMatch("*food*", wildcardStringMatchType, true);`
	`ObjectSearchResults results = session.getObjectBySearch(queries, os);`
	
	`// get new search inteface and reference previous result set`
	`os = session.getObjectSearch();`
	`os.searchWithinResults(results);`
	
	`queries[0].addDisplayNameMatch("pickles", wordStringMatchType, true);`
	`results = session.getObjectsBySearch(queries, os);`
	`OsidList pickles = results.getObjectList();`		

Method	**getResultSize**		
Description	Returns the size of a result set from a search query. This number serves as an estimate to provide feedback for refining search queries and may not be the number of elements available through an OsidList.		
Return	cardinal		the result size
Compliance	mandatory		This method must be implemented.
Method	**getProperties**		
Description	Gets a list of properties corresponding to the search type. Properties provide a means for applications to display a representation of the contents of an interface extension without understanding its Type specification. Applications needing to examine a specific property or perform updates should use the extension interface defined by its Type.		
Return	osid.PropertyList		a list of properties
Errors	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		an authorization failure occurred
Compliance	mandatory		This method must be implemented.
Method	**hasSearchRecordType**		
Description	Tests if this search results supports the given record Type. The given record type may be supported by the object through interface/type inheritence. This method should be checked before retrieving the record interface.		
Parameters	osid.type.Type	searchRecordType	a type
Return	boolean		true if a search record the given record Type is available, false otherwise
Errors	NULL_ARGUMENT		searchRecordType is null
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidSearchRecord**		
Implements			
Description	OsidSearchRecord is a top-level interface for all search record objects. A record is an auxiliary interface that can be retrieved from an OSID search or serach results that contains method definitions outside the core OSID specification. An OSID record interface specification is identified with a Type.		
Method	**implementsType**		
Description	Tests if the given type is implemented by this search record. Other types than that directly indicated by getType() may be supported through an inheritance scheme where the given type specifies a record that is a parent interface of the interface specified by ge		
Parameters	osid.type.Type	recordType	a type
Return	boolean		true if the given serach record Type is implemented by this record, false otherwise
Errors	NULL_ARGUMENT		recordType is null
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidReceiver**	
Implements		
Description	An OsidReceiver is used to receive asynchronous notifications from a service. The receiver defines the interface to be implemented by the consumer.	
Method	**up**	
Description	The callback for notifications that the notification bus is operational.	
Compliance	mandatory	This method must be implemented.
Method	**down**	
Description	The callback for notifications that the notification bus is not operating.	
Compliance	mandatory	This method must be implemented.

Interface	**osid.OsidList**	
Implements		
Description	OsidList is the top-level interface for all OSID lists. An OSID list provides sequential access, one at a time or many at a time, access to a set of elements. These elements are not required to be OsidObjects but generally are. The element retrieval methods are defined in the sub-interface of OsidList where the appropriate return type is defined. Osid lists are a once pass through iteration of elements. The size of the object set and the means in which the element set is generated or stored is not known. Assumptions based on the length of the element set by copying the entire contents of the list into a fixed buffer should be done with caution a awareness that an implementation may return a number of elements ranging from zero to infinity. Lists are returned by methods when multiple return values are possible. There is no guarantee that successive calls to the same method will return the same set of elements in a list. Unless an order is specified in an interface definition, the order of the elements is not known.	
Method	**hasNext**	
Description	Tests if there are more elements in this list.	
Return	boolean	true if more elements are available in this list, false if the end of the list has been reached
Compliance	mandatory	This method must be implemented.
Provider Notes	Any errors that may result from accesing the underlying set of elements are to be deferred until the consumer attempts retrieval in which case the provider must return true for this method.	
Method	**available**	
Description	Gets the number of elements available for retrieval. The number returned by this method may be less than or equal to the total number of elements in this list. To determine if the end of the list has been reached, the method hasNext() should be used. This method conveys what is known about the number of remaining elements at a point in time and can be used to determine a minimum size of the remaining elements, if known. A valid return is zero even if hasNext() is true. This method does not imply asynchronous usage. All OSID methods may block.	
Return	cardinal	the number of elements available for retrieval
Compliance	mandatory	This method must be implemented.
Provider Notes	Any errors that may result from accesing the underlying set of elements are to be deferred until the consumer attempts retrieval in which case the provider must return a positive integer for this method so the consumer can continue execution to receive the error. In all other circumstances, the provider must not return a number greater than the number of elements known since this number will be fed as a parameter to the bulk retrieval method.	
Method	**skip**	
Description	Skip the specified number of elements in the list. If the number skipped is greater than the number of elements in the list, hasNext() becomes false and available() returns zero as there are no more elements to retrieve.	
Parameters	cardinal	n — the number of elements to skip
Compliance	mandatory	This method must be implemented.

Interface	**osid.OsidCatalog**
Implements	osid.OsidObject
Description	OsidCatalog is the top level interface for all OSID catalog-like objects. A catalog relates to other OSID objects for the purpose of organization and federation. An example catalog is a Repository that relates to a collection of Assets. Catalogs allow for the retrieval of a provider identity and branding.

Interface	**osid.OsidCatalogQuery**		
Implements	osid.OsidQuery		
Description	The OsidCatalogQuery is used to assemble search queries for catalogs.		
Method	**matchProviderId**		
Description	Match the Id of the provider resource.		
Parameters	osid.id.Id	resourceId	id to match
	boolean	match	true if for a positive match, false for negative match
Errors	NULL_ARGUMENT		resourceId is null
Compliance	mandatory		This method must be implemented.
Method	**matchUnsetProvider**		
Description	Match assets with no provider.		
Compliance	mandatory		This method must be implemented.
Method	**supportsProviderQuery**		
Description	Tests if a ResourceQuery for the provider is available.		
Return	boolean		true if a resource query interface is available, false otherwise
Compliance	mandatory		This method must be implemented.
Method	**getProviderQuery**		
Description	Gets the query interface for the provider. Each retrieval performs a boolean OR.		
Parameters	boolean	match	true if for a positive match, false for negative match
Return	osid.resource.ResourceQuery		the provider query
Errors	UNIMPLEMENTED		supportsProviderQuery() is false
Compliance	optional		This method must be implemented if supportsProviderQuery() is true.

Interface	**osid.OsidCatalogForm**		
Implements	osid.OsidForm		
Description	This form is used to create and update catalogs.		
Method	**getProviderMetadata**		
Description	Gets the metadata for a provider.		
Return	osid.Metadata		metadata for the provider
Compliance	mandatory		This method must be implemented.
Method	**setProvider**		
Description	Sets a provider.		
Parameters	osid.id.Id	providerId	the new publisher
Errors	INVALID_ARGUMENT		providerId is invalid
	NO_ACCESS		Metadata.isReadOnly() is true
	NULL_ARGUMENT		providerId is null
Compliance	mandatory		This method must be implemented.
Method	**clearProvider**		
Description	Removes the provider.		
Errors	NO_ACCESS		Metadata.isRequired() is true or
Compliance	mandatory		This method must be implemented.

Interface	**osid.OsidCatalogSearchOrder**	
Implements	osid.OsidSearchOrder	
Description	An interface for specifying the ordering of catalog search results.	
Method	**orderByProvider**	
Description	Specifies a preference for ordering the results by provider. The element of the provider to order is not specified but may be managed through the provider ordering interface.	
Compliance	mandatory	This method must be implemented.
Method	**supportsProviderSearchOrder**	
Description	Tests if a ProviderSearchOrder interface is available.	
Return	boolean	true if a provider search order interface is available, false otherwise
Compliance	mandatory	This method must be implemented.
Method	**getProviderSearchOrder**	
Description	Gets the search order interface for a provider	
Return	osid.resource.ResourceSearchOrder	the provider search order interface
Errors	UNIMPLEMENTED	supportsProviderSearchOrder() is false
Compliance	optional	This method must be implemented if supportsProviderSearchOrder() is true.

Interface	**osid.OsidRuntimeProfile**	
Implements	osid.OsidProfile	
Description	The OsidRuntimeProfile defines the service aspects of the OSID runtime service.	
Method	**supportsConfiguration**	
Description	Tests if a configuration service is provided within this runtime environment.	
Return	boolean	true if a configuration service is available, false otherwise
Compliance	mandatory	This method must be implemented.
Method	**supportsInstallation**	
Description	Tests if an installation service is provided within this runtime environment.	
Return	boolean	true if a installation service is available, false otherwise
Compliance	mandatory	This method must be implemented.

Interface	**osid.OsidRuntimeManager**		
Implements	osid.OsidManager		
	osid.OsidRuntimeProfile		
Description	The OsidRuntimeManager represents and OSID platform and contains the information required for running OSID implementations such as search paths and configurations. The OsidRuntimeManager is defined as an interface to provide flexibility for managing an OSID environment. The instantiation of a OsidRuntimeManager implementation is defined by the OSID platform. The OsidRuntimeManager should be instantiated with a string that identifies the application or environment current at the time of instantiation. This key is used soley for the purpose of seeding the configuration service as a means to enable lower level OSIDs to tune their configuration in response to this key, or, it can be used by the application to retrieve configuration data for itself.		
Method	**getManager**		
Description	Finds, loads and instantiates providers of OSID managers. Providers must conform to an OsidManager interface. The interfaces are defined in the OSID enumeration. For all OSID requests, an instance of OsidManager that implements the OsidManager interface is returned. In bindings where permitted, this can be safely cast into the requested manager.		
Parameters	osid.OSID	osid	represents the OSID
	string	implClassName	the name of the implementation
	string	version	the minimum required interface version
Return	osid.OsidManager		the manager of the service
Errors	NOT_FOUND		the implementation class name was not found
	NULL_ARGUMENT		implClassName is null
	OPERATION_FAILED		unable to complete request
	UNSUPPORTED		implClassName does not support the requested OSID
Compliance	mandatory		This method must be implemented.
Provider Notes	After finding and instantiating the requested OsidManager, providers must invoke OsidManager.initialize(OsidRuntimeManager) where the environment is an instance of the current environment that includes the configuration for the service being initialized. The OsidRuntimeManager passed may include information useful for the configuration such as the identity of the service being instantiated.		

Method	getProxyManager		
Description	Finds, loads and instantiates providers of OSID managers. Providers must conform to an OsidManager interface. The interfaces are defined in the OSID enumeration. For all OSID requests, an instance of OsidManager that implements the OsidManager interface is returned. In bindings where permitted, this can be safely cast into the requested manager.		
Parameters	osid.OSID	osid	represents the OSID
	string	implementation	the name of the implementation
	string	version	the minimum required interface version
Return	osid.OsidProxyManager		the manager of the service
Errors	NOT_FOUND		the implementation package was not found
	NULL_ARGUMENT		implementation is null
	OPERATION_FAILED		unable to complete request
	UNSUPPORTED		implementation does not support the requested OSID
Compliance	mandatory		This method must be implemented.
Provider Notes	After finding and instantiating the requested OsidManager, providers must invoke OsidManager.initialize(OsidRuntimeManager) where the environment is an instance of the current environment that includes the configuration for the service being initialized. The OsidRuntimeManager passed may include information useful for the configuration such as the identity of the service being instantiated.		

Method	getConfiguration	
Description	Gets the current configuration in the runtime environment.	
Return	osid.configuration.ValueLookupSession	a configuration
Errors	OPERATION_FAILED	unable to complete request
	PERMISSION_DENIED	an authorization failure occured
	UNIMPLEMENTED	a configuration service is not supported
Compliance	optional	This method must be implemented if supportsConfiguration() is true.

Method	getConfigurationManager	
Description	Gets the current configuration for updating in the runtime environment.	
Return	osid.configuration.ConfigurationManager	a configuration manager
Errors	OPERATION_FAILED	unable to complete request
	UNIMPLEMENTED	a configuration service is not supported
Compliance	optional	This method must be implemented if supportsConfiguration() is true.
Provider Notes	A configuration service may provide user-specific configurations by making use of an authentication service.	

Method	getInstallationManager	
Description	Gets the installation manager used in the runtime environment.	
Return	osid.installation.InstallationManager	a configuration manager
Errors	OPERATION_FAILED	unable to complete request
	UNIMPLEMENTED	a configuration service is not supported
Compliance	optional	This method must be implemented if supportsInstallation() is true.

Enumeration	**osid.OSID**	
Description	This enumeration contains the list of OSIDs.	
Values	ASSESSMENT	The Assessment Open Service Interface Definition.
	AUTHENTICATION	The Authentication Open Service Interface Definition.
	AUTHORIZATION	The Authorization Open Service Interface Definition.
	CATALOGGING	The Catalogging Open Service Interface Definition.
	CONFIGURATION	The Configuration Open Service Interface Definition.
	COURSE	The Course Open Service Interface Definition.
	DICTIONARY	The Dictionary Open Service Interface Definition.
	FILING	The Filing Open Service Interface Definition.
	GRADING	The Grading Open Service Interface Definition.
	HIERARCHY	The Hierarchy Open Service Interface Definition.
	ID	The Id Open Service Interface Definition.
	INSTALLATION	The Installation Open Service Interface Definition.
	LOCALE	The Locale Open Service Interface Definition.
	LOGGING	The Logging Open Service Interface Definition.
	MESSAGING	The Messaging Open Service Interface Definition.
	PROVISIONING	The Provisioning Open Service Interface Definition.
	REPOSITORY	The Repository Open Service Interface Definition.
	RESOURCE	The Resource Open Service Interface Definition.
	SCHEDULING	The Scheduling Open Service Interface Definition.
	TOPOLOGY	The Topology Open Service Interface Definition.
	TRANSACTION	The Transaction Open Service Interface Definition.
	TRANSPORT	The Transport Open Service Interface Definition.
	TYPE	The Type Open Service Interface Definition.
	WORKFLOW	The Workflow Open Service Interface Definition.

Interface	**osid.Metadata**	
Implements		
Description	The Metadata interface defines a set of methods describing a the syntax and rules for updating a data element or property inside an OSID object. This interface provides a means to retrieve special restrictions placed upon data elements such as sizes and ranges that may vary from provider to provider or from object to object.	
Method	**getInstructions**	
Description	Gets instructions for updating this data. This is a human readable description of the data element or property that may include special instructions or caveats to the end-user above and beyond what this interface provides.	
Return	string	instructions
Compliance	mandatory	This method must be implemented.
Method	**isRequired**	
Description	Tests if this data element is required for creating new objects.	
Return	boolean	true if this data is required, false otherwise
Compliance	mandatory	This method must be implemented.
Method	**isSet**	
Description	Tests if this data element is has a value.	
Return	boolean	true if this data has been set, false otherwise
Compliance	mandatory	This method must be implemented.
Method	**isReadOnly**	
Description	Tests if this data can be updated. This may indicate the result of a pre-authorization but is not a guarantee that an authorization failure will not occur when the create or update transaction is issued.	
Return	boolean	true if this data is not updatable, false otherwise
Compliance	mandatory	This method must be implemented.
Method	**getSyntax**	
Description	Gets the syntax of this data.	
Return	osid.MetadataSyntax	an enumeration indicating the type of value
Compliance	mandatory	This method must be implemented.
Method	**getUnits**	
Description	Gets the units of this data for display purposes ('lbs', 'gills', 'furlongs').	
Return	string	the display units of this data or an empty string if not applicable
Compliance	mandatory	This method must be implemented.
Method	**getMinCardinal**	
Description	Gets the minimum cardinal value.	
Return	cardinal	the minimum value
Errors	ILLEGAL_STATE	syntax is not a CARDINAL
Compliance	mandatory	This method must be implemented.
Method	**getMaxCardinal**	
Description	Gets the maximum cardinal value.	
Return	cardinal	the maximum value
Errors	ILLEGAL_STATE	syntax is not a CARDINAL
Compliance	mandatory	This method must be implemented.
Method	**getCardinalSet**	
Description	Gets the set of acceptable cardinal values.	
Return	cardinal[]	the set of values
Errors	ILLEGAL_STATE	syntax is not a CARDINAL
Compliance	mandatory	This method must be implemented.

Method	getMinDateTime	
Description	Gets the minimum date value.	
Return	osid.calendaring.DateTime	the minimum value
Errors	ILLEGAL_STATE	syntax is not a DATETIME
Compliance	mandatory	This method must be implemented.
Method	getMaxDateTime	
Description	Gets the maximum date value.	
Return	osid.calendaring.DateTime	the maximum value
Errors	ILLEGAL_STATE	syntax is not a DATETIME
Compliance	mandatory	This method must be implemented.
Method	getDateTimeSet	
Description	Gets the set of acceptable date time values.	
Return	osid.calendaring.DateTime[]	the set of values
Errors	ILLEGAL_STATE	syntax is not a DATETIME
Compliance	mandatory	This method must be implemented.
Method	getDateTimeResolution	
Description	Gets the resolution of the date time value.	
Return	osid.calendaring.DateTimeResolution	the resolution
Errors	ILLEGAL_STATE	syntax is not a DATETIME
Compliance	mandatory	This method must be implemented.
Method	getMinFloat	
Description	Gets the minimum float value.	
Return	float	the minimum value
Errors	ILLEGAL_STATE	syntax is not a FLOAT
Compliance	mandatory	This method must be implemented.
Method	getMaxFloat	
Description	Gets the maximum float value.	
Return	float	the maximum float
Errors	ILLEGAL_STATE	syntax is not a FLOAT
Compliance	mandatory	This method must be implemented.
Method	getFloatSet	
Description	Gets the set of acceptable float values.	
Return	float[]	the set of values
Errors	ILLEGAL_STATE	syntax is not a FLOAT
Compliance	mandatory	This method must be implemented.
Method	getMinInteger	
Description	Gets the minimum integer value.	
Return	integer	the minimum value
Errors	ILLEGAL_STATE	syntax is not an INTEGER
Compliance	mandatory	This method must be implemented.
Method	getMaxInteger	
Description	Gets the maximum integer value.	
Return	integer	the maximum value
Errors	ILLEGAL_STATE	syntax is not an INTEGER
Compliance	mandatory	This method must be implemented.
Method	getIntegerSet	
Description	Gets the set of acceptable integer values.	
Return	integer[]	the set of values
Errors	ILLEGAL_STATE	syntax is not an INTEGER
Compliance	mandatory	This method must be implemented.

Method	getMinStringSize	
Description	Gets the minimum string size.	
Return	cardinal	the minimum string length
Errors	ILLEGAL_STATE	syntax is not a STRING
Compliance	mandatory	This method must be implemented.

Method	getMaxStringLength	
Description	Gets the maximum string length.	
Return	cardinal	the maximum string length
Errors	ILLEGAL_STATE	syntax is not a STRING
Compliance	mandatory	This method must be implemented.

Method	getStringSet	
Description	Gets the set of acceptable string values.	
Return	string[]	the set of values
Errors	ILLEGAL_STATE	syntax is not a STRING
Compliance	mandatory	This method must be implemented.

Method	getIdSet	
Description	Gets the set of acceptable Ids.	
Return	osid.id.Id[]	the set of Ids
Errors	ILLEGAL_STATE	syntax is not an ID
Compliance	mandatory	This method must be implemented.

Method	getTypeSet	
Description	Gets the set of acceptable Types.	
Return	osid.type.Type[]	the set of Types
Errors	ILLEGAL_STATE	syntax is not an TYPE
Compliance	mandatory	This method must be implemented.

Enumeration	**osid.MetadataSyntax**	
Description	This enumeration contains the possible value types.	
Vaues	NONE	No value available.
	BOOLEAN	A truth value of true or false.
	CARDINAL	A non-negative number supporting a 64-bit value (0..9,223,372,036,854,775,808). Cardinal numbers should be used to represent numbers such as sizes and counters where negative numbers have no meaning.
	DATETIME	An OSID DateTime.
	FLOAT	A signed floating point number supporting a signed significand of range -281,474,976,710,656.. 281,474,976,710,656 and an 8-bit exponent (1..255).
	ID	An OSID Id.
	INTEGER	A number supporting a 64-bit value (-9,223,372,036,854,775,808.. 9,223,372,036,854,775,808).
	OBJECT	An arbitrary object.
	STRING	A string of characters.
	TYPE	An OSID Type.

Interface	**osid.SpatialUnit**		
Implements			
Description	The SpatialUnit interface defines a point or region in space. The domain indicates the spatial corrdinate system that maps to an interface specifcation of its type.		
Method	**implementsDomainType**		
Description	Tests if the diven domain is available for this spatial unit.		
Parameters	osid.type.Type	domainType	the domain type
Return	boolean		true if the given domain type is supported, false otherwise
Compliance	mandatory		This method must be implemented.
Method	**getDomainType**		
Description	Gets the domain type for this spatial unit. supportsDomain() should be used to test for interoperability.		
Return	osid.type.Type		the domain type
Compliance	mandatory		This method must be implemented.
Method	**isInclusive**		
Description	Tests if the given spatial unit is completely included in this one.		
Parameters	osid.SpatialUnit	spatialUnit	the spatial unit to compare
Return	boolean		true if the given spatial unit is included in this one, false otherwise
Errors	NULL_ARGUMENT		spatialUnit is null
	UNSUPPORTED		spatialUnit is not supported
Compliance	mandatory		This method must be implemented.
Method	**isExclusive**		
Description	Tests if the given spatial unit is completely exclusive in this one.		
Parameters	osid.SpatialUnit	spatialUnit	the spatial unit to compare
Return	boolean		true if the given spatial unit is exclsuive of this one, false otherwise
Errors	NULL_ARGUMENT		spatialUnit is null
	UNSUPPORTED		spatialUnit is not supported
Compliance	mandatory		This method must be implemented.
Method	**isEqual**		
Description	Tests if the given spatial unit is equal to this one.		
Parameters	osid.SpatialUnit	spatialUnit	the spatial unit to compare
Return	boolean		true if the given spatial unit is equal to this one, false otherwise
Errors	NULL_ARGUMENT		spatialUnit is null
	UNSUPPORTED		spatialUnit is not supported
Compliance	mandatory		This method must be implemented.
	getSpatialDomain		
Description	Gets the typed interface corresponding to this Spatial domain.		
Parameters	osid.type.Type	domainType	the domain type
Return	osid.SpatialUnit		the spatial domain with the typed interface
Errors	NULL_ARGUMENT		domainType is null
	UNSUPPORTED		supportsDomainType(domainType) is false
Compliance	mandatory		This method must be implemented.

	getSpatialUnitExtension		
Description	Gets the typed interface corresponding to this SpatialUnit Type.		
Parameters	osid.type.Type	domainType	the spatial unit domain type
Return	osid.SpatialUnit		the spatial unit with the typed interface
Errors	NULL_ARGUMENT		domainType is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure occurred
	UNSUPPORTED		implementsDomainType(domainType) is false
Compliance	mandatory		This method must be implemented.

Interface	**osid.SpatialUnitList**		
Implements	osid.OsidList		

Description	Like all OsidLists, SpatialUnitList provides a means for accessing SpatialUnit elements sequentially either one at a time or many at a time. Examples:
	
	`while (sull.hasNext()) {`
	`  SpatialUnit su = sul.getNextSpatialUnit();`
	`}`
	
	or
	
	`while (sul.hasNext()) {`
	`  SpatialUnit[] sus = sul.getNextSpatialUnit(sul.available());`
	`}`		

Method	**getNextSpatialUnit**		
Description	Gets the next SpatialUnit in this list.		
Return	osid.SpatialUnit		the next SpatialUnit in this list. The hasNext() method should be used to test that a next SpatialUnit is available before calling this method.
Errors	ILLEGAL_STATE		no more elements available in this list
	OPERATION_FAILED		unable to complete request
Compliance	mandatory		This method must be implemented.
Method	**getNextSpatialUnits**		
Description	Gets the next set of SpatialUnit elements in this list which must be less than or equal to the number returned from available().		
Parameters	cardinal	n	the number of SpatialUnit elements requested which should be less than or equal to available()
Return	osid.SpatialUnit[]		an array of SpatialUnit elements. The length of the array is less than or equal to the number specified.
Errors	ILLEGAL_STATE		no more elements available in this list
	OPERATION_FAILED		unable to complete request
Compliance	mandatory		This method must be implemented.

Interface	**osid.Property**	
Implements		
Description	A Property is a representation of data in string form. Properties are exposed in OSID objects as a means of providing a quick gestalt of data elements whose specifics are described through a type specification. A view of an OSID object via Properties allows applications to browse the content without understanding the type specification in place, but any acquisition of specific data, access to an object or other primitive type, or changing the data requires the typed interfaces.	
Method	**getDisplayName**	
Description	The display name for this property.	
Return	string	the display name
Compliance	mandatory	This method must be implemented.
Method	**getDisplayLabel**	
Description	A short display label.	
Return	string	the display label
Compliance	mandatory	This method must be implemented.
Method	**getDescription**	
Description	A description of this property.	
Return	string	the description
Compliance	mandatory	This method must be implemented.
Method	**getValue**	
Description	The value of this property.	
Return	string	the value
Compliance	mandatory	This method must be implemented.

Interface	**osid.PropertyList**		
Implements	osid.OsidList		

Description	Like all OsidLists, PropertyList provides a means for accessing Property elements sequentially either one at a time or many at a time. Examples:
	
	`while (pl.hasNext()) {`
	`  Property property = pl.getNextProperty();`
	`}`
	
	or
	
	`while (pl.hasNext()) {`
	`  Property[] properties = pl.getNextProperties(pl.available());`
	`}`		

Method	**getNextProperty**		
Description	Gets the next Property in this list.		
Return	osid.Property		the next Property in this list. The hasNext() method should be used to test that a next Property is available before calling this method.
Errors	ILLEGAL_STATE		no more elements available in this list
	OPERATION_FAILED		unable to complete request
Compliance	mandatory		This method must be implemented.
Method	**getNextProperties**		
Description	Gets the next set of Property elements in this list which must be less than or equal to the number returned from available().		
Parameters	cardinal	n	the number of Property elements requested which should be less than or equal to available()
Return	osid.Property[]		an array of Property elements. The length of the array is less than or equal to the number specified.
Errors	ILLEGAL_STATE		no more elements available in this list
	OPERATION_FAILED		unable to complete request
Compliance	mandatory		This method must be implemented.

Interface	**osid.ServiceReceiver**		
Implements			

Description	A ServiceReceiver is used to receive asynchronous notifications from a service. The receiver defines the interface to be implemented by the consumer. Simple example:
	
	`MyCallback {`
	`  void up() {`
	`  print "notification service is up";`
	`  }`
	
	`  void down() {`
	`  print "notification service is down";`
	`  }`
	
	`  void newMessage(msg) {`
	`  print ("new message received " + msg);`
	`  }`
	`}`
	
	`notificationSession = manager.registerForServiceMessages(myCallback);`		

Method	**newMessage**		
Description	The callback for notifications of new messages. The message Id is to eliminate duplicate messages that may originate from shared providers.		
Parameters	osid.id.Id	messageId	unique identifier for the message
	string	message	a service message
Compliance	mandatory		This method must be implemented.