Open Knowledge Initiative

OSID V3 Specifications
type 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 © 2002-2007 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.type package

Interfaces	osid.type.TypeProfile osid.type.TypeManager osid.type.TypeProxyManager osid.type.TypeLookupSession osid.type.TypeAdminSession osid.type.Type osid.type.TypeList

Package	**osid.type**
Title	**Type Open Service Interface Definitions**
Version	3.0.0

Description	The type package defines a set of interfaces for managing Type definitions. Types are used as an identifier primarily for identification of interface extensions throughout the OSIDs and occasionally used as an extensible enumeration. An agreement between a consumer and a provider means they support the same Type.
	
	A Type is similar to an Id but includes other data for display and organization. The identification portion of the Type is globally unique and contains:
	• authority: the name of the entity or organization responsible for the type. Using a domain name is a reasonable convention.
	• identifier: a string serving as an id. The identifier may be a urn, guid, oid or some other means of identification. Since all of the identification elements including the domain and authority create an overall unique Type, the identifier may even be a sequence number defined within a particular domain.
	• namespace: a string identifying the namespace of the identifier, such as "urn" or "oid".
	
	Lookup example:
	
	`Type type = lookupSession.getType("asset", "uri",`
	`                                  "http://harvestroad.com/osidTypes/image",`
	`                                  "harvestroad.com");`
	`print type.getDisplayName();`
	
	The sessions in this OSID offer the capabilities of a Type registry to centrally manage definitions and localized display strings and descriptions. Applications may opt to construct their own Types directly and bypass this service.
	Types are part of an internal hierarchy. The type hierarchy generally parallels the interface hierarchy in an OSID object. For example, an Asset may offer an interface extension for a jpeg image whose definition extends a more generic image interface. The type hierarchy may look like assetType -> assetImageType -> assetJPEGType. Note that the root of the hierarchy is the core object interface. Since the types are part of a specification in itself, the Type contains knowledge if its own hierarchy.
	
	It is possible to bypass the Type OSID entirely and simply construct the type classes directly. This has the feature of being expedient and contains the type definitions with their consuming code. This may get cumbersome for managing larger numbers of types especially when they come in the form of hierarchies.
	
	Unless an application will display a type, it can simply construct a type based on the identification components and not use this service. OSID implementations benefit more by using this service since the type hierarchy is necessary in order to respond to interoperability tests and it, provides a place to perform mappings across different type definitions, and provides displayable metadata to its consumers.
	
	Most OSID interfaces are used to encapsulate implementation-specific objects from provider to consumer. The Type interface is bi-directional and as such cannot be used to encapsulate implementation-specific data other than what is defined explicitly in the Type. A provider must respect any Type based on its interface alone.

Interface	**osid.type.TypeProfile**	
Implements	osid.OsidProfile	
Description	The TypeProfile describes the interoperability among type services.	
Method	**supportsTypeLookup**	
Description	Tests if Type lookup is supported.	
Return	boolean	true if Type lookup is supported, false otherwise
Compliance	mandatory	This method must be implemented.
Method	**supportsTypeAdmin**	
Description	Tests if a Type administrative service is supported.	
Return	boolean	true if Type administration is supported, false otherwise
Compliance	mandatory	This method must be implemented.

Interface	**osid.type.TypeManager**	
Implements	osid.OsidManager	
	osid.type.TypeProfile	
Description	This manager provides access to the available sessions of the type service. The TypeLookupSession is used for looking up Types and the TypeAdminSession is used for managing and registering new Types.	
Method	**getTypeLookupSession**	
Description	Gets the OsidSession associated with the type lookup service.	
Return	osid.type.TypeLookupSession	a TypeLookupSession
Errors	OPERATION_FAILED	unable to complete request
	UNIMPLEMENTED	supportsTypeLookup() is false
Compliance	optional	This method must be implemented if supportsTypeLookup() is true.
Method	**getTypeAdminSession**	
Description	Gets the OsidSession associated with the type admin service.	
Return	osid.type.TypeAdminSession	a TypeAdminSession
Errors	OPERATION_FAILED	unable to complete request
	UNIMPLEMENTED	supportsTypeAdmin() is false
Compliance	optional	This method must be implemented if supportsTypeAdmin() is true.

Interface	**osid.type.TypeProxyManager**		
Implements	osid.OsidProxyManager		
	osid.type.TypeProfile		
Description	This manager provides access to the available sessions of the type service. Methods in this manager support the passing of an Authentication object for the purpose of proxy authentication. The TypeLookupSession is used for looking up Types and the TypeAdminSession is used for managing and registering new Types.		
Method	**getTypeLookupSession**		
Description	Gets the OsidSession associated with the TypeBrowser service using the supplied Authentication.		
Parameters	osid.authentication.Authentication	authentication	proxy authentication
Return	osid.type.TypeLookupSession		a TypeLookupSession
Errors	NULL_ARGUMENT		authentication is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authentication is invalid
	UNIMPLEMENTED		supportsTypeLookup() is false
	UNSUPPORTED		the authentication service is not supported
Compliance	optional		This method must be implemented if supportsTypeLookup() is true.
Method	**getTypeAdminSession**		
Description	Gets the OsidSession associated with the TypeAdmin service and supplied Authentication.		
Parameters	osid.authentication.Authentication	authentication	proxy authentication
Return	osid.type.TypeAdminSession		the new TypeAdminSession
Errors	NULL_ARGUMENT		authentication is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authentication is invalid
	UNIMPLEMENTED		supportsTypeAdmin() is false
	UNSUPPORTED		the authentication service is not supported
Compliance	optional		This method must be implemented if supportsTypeAdmin() is true.

Interface	**osid.type.TypeLookupSession**		
Implements	osid.OsidSession		
Description	This session retrieves Types. A single Type can be retrieved using getType() and all types known to this service can be accessed via getTypes().		
Method	**canLookupTypes**		
Description	Tests if this user can perform Type lookups. A return of true does not guarantee successful authorization. A return of false indicates that it is known all methods in this session will result in a PERMISSION_DENIED. This is intended as a hint to an application that may opt not to offer lookup operations.		
Return	boolean		false if lookup methods are not authorized, true otherwise
Compliance	mandatory		This method must be implemented.
Method	**getType**		
Description	Gets a Type by its string representation which is a combination of the authority and identifier. This method only returns the Type if it is known by the given identification components.		
Parameters	string	namespace	the identifier namespace
	string	identifier	the identifier
	string	authority	the authority
Return	osid.type.Type		the Type
Errors	NOT_FOUND		the type is not found
	NULL_ARGUMENT		null argument provided
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.
Method	**hasType**		
Description	Tests if the given Type is known.		
Parameters	osid.type.Type	type	the Type to look for
Return	boolean		true if the given Type is known, false otherwise
Errors	NULL_ARGUMENT		type is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.
Method	**getTypesByDomain**		
Description	Gets all the known Types by domain.		
Parameters	string	domain	the domain
Return	osid.type.TypeList		the list of Types with the given domain
Errors	NULL_ARGUMENT		domain is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.
Method	**getTypesByAuthority**		
Description	Gets all the known Types by authority.		
Parameters	string	authority	the authority
Return	osid.type.TypeList		the list of Types with the given authority
Errors	NULL_ARGUMENT		authority is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		respect my authoritay
Compliance	mandatory		This method must be implemented.

Method	getTypesByDomainAndAuthority		
Description	Gets all the known Types by domain and authority		
Parameters	string	domain	the domain
	string	authority	the authority
Return	osid.type.TypeList		the list of Types with the given domain and authority
Errors	NULL_ARGUMENT		domain or authority is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.

Method	getTypes	
Description	Gets all the known Types.	
Return	osid.type.TypeList	the list of all known Types
Errors	OPERATION_FAILED	unable to complete request
	PERMISSION_DENIED	authorization failure
Compliance	mandatory	This method must be implemented.

Interface	**osid.type.TypeAdminSession**		
Implements	osid.OsidSession		
Description	This session is used to create, update and delete Types in the registry.		
Method	**canCreateTypes**		
Description	Tests if this user can create Types. A return of true does not guarantee successful authorization. A return of false indicates that it is known creating a Type will result in a PERMISSION_DENIED. This is intended as a hint to an application that may opt not to offer create operations to an unauthorized user.		
Return	boolean		false if Type creation is not authorized, true otherwise
Compliance	mandatory		This method must be implemented.
Method	**getTypeForm**		
Description	Gets the type form for creating new types. A new form should be requested for each create transaction.		
Return	osid.type.TypeForm		the type form
Compliance	mandatory		This method must be implemented.
Method	**createType**		
Description	Creates a new Type.		
Parameters	string	authority	the authority
	string	identifierNS	the namespace of the identifier
	string	identifier	the identifier
	osid.type.TypeForm	typeForm	the type form
Return	osid.type.Type		the created Type
Errors	INVALID_ARGUMENT		one or more of the arguments is invalid
	NULL_ARGUMENT		one or more of the arguments is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.
Method	**canUpdateTypes**		
Description	Tests if this user can update types. A return of true does not guarantee successful authorization. A return of false indicates that it is known updating a Type will result in a PERMISSION_DENIED. This is intended as a hint to an application that may opt not to offer update operations to an unauthorized user.		
Return	boolean		false if type modification is not authorized, true otherwise
Compliance	mandatory		This method must be implemented.
Method	**updateType**		
Description	Updates a display name.		
Parameters	osid.type.Type	type	the Type to be updated
	osid.type.TypeForm	typeForm	the type form
Errors	INVALID_ARGUMENT		displayName or displayLabel is not valid
	NOT_FOUND		type is not found
	NULL_ARGUMENT		type, displayName or displayLabel is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.

Method	**canDeleteTypes**		
Description	Tests if this user can delete Types from this ItemBank. A return of true does not guarantee successful authorization. A return of false indicates that it is known deleting a Type will result in a PERMISSION_DENIED. This is intended as a hint to an application that may opt not to offer delete operations to an unauthorized user.		
Return	boolean		false if Item deletion is not authorized, true
Compliance	mandatory		This method must be implemented.
Method	**deleteType**		
Description	Removes a Type.		
Parameters	osid.type.Type	type	the Type to remove
Errors	NOT_FOUND		type is not found
	NULL_ARGUMENT		type is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.
Method	**addChild**		
Description	Adds a type as a child to another.		
Parameters	osid.type.Type	parentType	the parent Type
	osid.type.Type	type	the type to add to parentType
Errors	ALREADY_EXISTS		type is already a child of parentType
	NOT_FOUND		type or parentType is not found
	NULL_ARGUMENT		type or parentType is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.
Method	**removeChild**		
Description	Removes a child from a type. This method removes the child but does not delete the type.		
Parameters	osid.type.Type	parentType	the parent Type
	osid.type.Type	type	the type to to remove
Errors	NOT_FOUND		type or parentType is not found
	NULL_ARGUMENT		type or parentType is null
	OPERATION_FAILED		unable to complete request
	PERMISSION_DENIED		authorization failure
Compliance	mandatory		This method must be implemented.

Interface	**osid.type.Type**	
Implements		
Description	The Type is a form of identifier that is primarily used to identify interface specifications. The Type differs from Id in that it offers display information and Types may be arranged in hierarchies to indicate an extended interface. Semantically, an Id identifies any OSID object while the Type identifies a specification. The components of the Type that make up its identification are: • identifier: a unique key or guid • namespace: the namespace of the identifier • authority: the isuer of the identifier Persisting a type reference means to persist the above identification elements. In addition to these identifier components, A Type mai also provide some additional metadata such as a name, description and domain.	
Method	**getDisplayName**	
Description	Gets the full display name of this Type.	
Return	string	the display name of this Type
Compliance	mandatory	This method must be implemented.
Method	**getDisplayLabel**	
Description	Gets the shorter display label for this Type. Where a display name of a Type might be "Critical Logging Priority Type", the display label could be "critical".	
Return	string	the display label for this Type. The display name is
Compliance	mandatory	This method must be implemented.
Method	**getDescription**	
Description	Gets a description of this Type.	
Return	string	the description of this Type. An empty string is returned when no description is available for this Type.
Compliance	mandatory	This method must be implemented.
Method	**getDomain**	
Description	Gets the domain. The domain can provide an information label about ths application space of this Type.	
Return	string	the domain of this Type
Compliance	mandatory	This method must be implemented.
Method	**getAuthority**	
Description	Gets the authority of this Type. The authority is a string used to ensure the uniqueness of this Type when using a non-federated identifier space. Generally, it is a domain name identifying the party responsible for this Type. This method is used to compare one Type to another.	
Return	string	the authority of this Type
Compliance	mandatory	This method must be implemented.
Method	**getIdentifierNamespace**	
Description	Gets the namespace of the identifier. This method is used to compare one Type to another.	
Return	string	the authority of this Type
Compliance	mandatory	This method must be implemented.
Method	**getIdentifier**	
Description	Gets the identifier of this Type. This method is used to compare one Type to another.	
Return	string	the identifier of this Type
Compliance	mandatory	This method must be implemented.

Method	isEqual		
Description	Determines if the given Type is equal to this one. Two Types are equal if the authority, namespace and identifier components are equal. The identifier is case sensitive while the authority strings are not case sensitive.		
Parameters	osid.type.Type	type	the Type to compare
Return	boolean		true if the given Type is equal to this one, false otherwise
Compliance	mandatory		This method must be implemented.
Method	isParent		
Description	Tests if the given type is a parent of this type.		
Parameters	osid.type.Type	type	the Type
Return	boolean		true if type is a parent of this type, false otherwise
Errors	NULL_ARGUMENT		type is null
Compliance	mandatory		This method must be implemented.
Method	isAncestor		
Description	Tests if the given type is an ancestor of this type.		
Parameters	osid.type.Type	type	the Type
Return	boolean		true if type is an ancestor of this type, false otherwise
Errors	NULL_ARGUMENT		type is null
Compliance	mandatory		This method must be implemented.
Method	getParents		
Description	Gets the parents of this type.		
Return	osid.type.TypeList		the parents of this type
Compliance	mandatory		This method must be implemented.

Interface	**osid.type.TypeForm**		
Implements			
Description	This form provides a means of updating various fields in the Type. Note that the domain, authority and identifier are part of the Type identification, and as such not modifiable.		
Method	**getDisplayNameMetadata**		
Description	Gets the metadata for the 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	**getDisplayLabelMetadata**		
Description	Gets the metadata for the display label.		
Return	osid.Metadata		metadata for the display label
Compliance	mandatory		This method must be implemented.
Method	**setDisplayLabel**		
Description	Seta a display label.		
Parameters	string	displayLabel	the new display label
Errors	INVALID_ARGUMENT		displayLabel is invalid
	NO_ACCESS		displayLabel cannot be modified
	NULL_ARGUMENT		displayLabel is null
Compliance	mandatory		This method must be implemented.
Method	**getDescriptionMetadata**		
Description	Gets the metadata for the 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	**getDomainMetadata**		
Description	Gets the metadata for the domain.		
Return	osid.Metadata		metadata for the domain
Compliance	mandatory		This method must be implemented.

Method	setDomain		
Description	Sets a domain.		
Parameters	string	domain	the new domain
Errors	INVALID_ARGUMENT		domain is invalid
	NO_ACCESS		domain cannot be modified
	NULL_ARGUMENT		domain is null
Compliance	mandatory		This method must be implemented.

Method	clearDomain	
Description	Clears the domain.	
Errors	NO_ACCESS	domain cannot be modified
Compliance	mandatory	This method must be implemented.

Interface	**osid.type.TypeList**		
Implements	osid.OsidList		

Description	Like all OsidLists, TypeList provides a means for accessing Type elements sequentially either one at a time or many at a time. Examples:
	
	`while (tl.hasNext()) {`
	`     Type type = tl.getNextType();`
	`}`
	
	or
	
	`while (tl.hasNext()) {`
	`     Type[] types = tl.getNextTypes(tl.available());`
	`}`		

Method	**getNextType**		
Description	Gets the next Type in this list.		
Return	osid.type.Type		the next Type in this list. The hasNext() method should be used to test that a next Type 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	**getNextTypes**		
Description	Gets the next set of Types in this list. The specified amount must be less than or equal to the return from available().		
Parameters	cardinal	n	the number of Type elements requested which must be less than or equal to available()
Return	osid.type.Type[]		an array of Type 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.