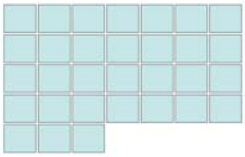
Open Service Interface Definition

Scheduling



Document Release: 2.0
OSID Release: 2.0

Summary

The Scheduling Open Service Interface Definition (OSID) provides a means of associating Agents with specific activities (ScheduleItems) that have a specific start and end date and time. For each ScheduleItem, the Agent or Agents involved have a Status that reflects their level of commitment to the activity. There are provisions for both recurring ScheduleItems and for finding the times when a set of Agents are available.

The Scheduling OSID is referenced in the Term of the CourseManagement OSID. The OSID provides a way for an application to integrate or use an external calendar system. In this way an application could provide calendar functionality, while still allowing integration with an Enterprise calendar system.

Service Definition

An examination of an Open Service Interface Definition (OSID) usually begins with the Manager. All Managers[1] provide the way to create the objects that implement the principal interfaces in the Service. Before discussing the Manager in detail, we will review the intended role of the Scheduling Service overall.

Scheduling serves to record information. An application might call an implementation of the Service to record user activity of any kind. For example, application use statistics such as who was using the application and what they did, which application features were invoked, exceptions raised by implementations and handled by the application, and so on. This might all be information about a system in production. In this mode, logs might be written as well as read. Additionally, an implementation of an OSID or anything else might want to log information for debugging, problem identification, or performance data. In this mode, logs might only be written.

[1] org.osid.OsidManager defines the interface extended by the Managers in each OSID. Here we will be discussing org.osid.scheduling.SchedulingManager. Refer to OsidManager for more information.

There are a variety of commercial Scheduling applications and there is Scheduling support in certain operating systems and development kits. OSID Scheduling is an abstraction that seeks to address a wide range of needs. The Service is general and focuses on reading from and writing to logs only.

org.osid.Scheduling.SchedulingManager

An installation may have several implementations of SchedulingManager available. Like managers in others O.K.I. services, SchedulingManagers with desired features can be obtained through OsidLoader. Once a SchedulingManager is available, it can create ScheduleItems. The Item has a start and end date and time, a set of participants, and optionally a master identifier such as a common value for recurring Items. The start and end date and time and the participants are mutable. The **createScheduleItem**(), **deleteScheduleItem**(), **getScheduleItem**(), and several **getScheduleItems**() methods support managing and interrogating these Items.

The Service provides for asking an implementation for when a set of Agents are available. This **getAvailableTimes**() method can be used before creating a ScheduleItem to find a time when all parties can attend. This also can be used when rescheduling an Item, perhaps after someone's commitment Status reflects that they cannot attend at the currently proposed time.

Method Summary

ShceduleItem	createScheduleItem(String displayName, String description, Id[] agents, long start, long end, String masterIdentifier)
void	deleteScheduleItem(Id scheduleItemId)
TimespanIterator	getAvailableTimes(Id[] agents, long start, long end)
TypeIterator	getItemCommitmentStatusTypes()
TypeIterator	getItemStatusTypes()
ScheduleItem	getScheduleItem(Id scheduleItemId)
ScheduleItemIterator	getScheduleItems(long start, long end, Type status)
ScheduleItemIterator	getScheduleItemsForAgents(long start, long end, Type status, Id[] agents)
ScheduleItemIterator	getScheduleItemsByMasterId(String masterIdentifier)

org.osid.scheduling.ScheduleItem

A ScheduleItem captures who is involved in the Item, when the Item occurs, and optionally a reference to how this Item relates to others. When the Item is created through the Manager, there is an AgentCommitment for each Agent specified. Additional Agents can be added after the Item exists and the Agents Status, that is their level of commitment to the Item, can be changed. The **addAgentCommitment**() and **changeAgentCommitment**(), and **getAgentCommitments**() method supporting managing the list of participants and their Status. The getCreator() method returns the Agent that created the ScheduleItem.

Each ScheduleItem has a unique Id. When the Item is created, this Id is generated by the implementation. When creating an Item, an application can optionally provide a master identifier. This identifier might tie all recurring Items together. The identifier might be an Id, but it also might be a rule for the recurrence relationship, or anything else. The unique Id and master identifier are returned by **getId**() and **getMasterIdentifier**(), respectively.

The start and end date and time for the Item is mutable. The **getStart**(), **getEnd**(), **updateStart**(), and **updateEnd**() methods relate to this. The Item can also have a Status such as "past", "tentative", "over subscribed", etc.

Method Summary

void	addAgentCommitment(Id agentId, Type agentStatus)
void	changeAgentCommitment(Id agentId, Type agentStatus)
AgentCommitmentIterator	getAgentCommitments()
Id	getCreator()
String	getDescription()
String	getDisplayName()
long	getEnd()
Id	getId()
String	getMasterIdentifier()
TypeIterator	getPropertyTypes()
Properties	getPropertiesByType(Type propertiesType)
PropertiesIterator	getProperties()
Type	getStatus()
long	getStart()
void	updateDescription(String description)
void	updateDisplayName(String displayName)
void	updateEnd(long end)
void	updateStart(long start)
void	updateStatus(Type status)

org.osid.scheduling.AgentCommitment

The AgentCommitment captures the level of commitment of an Agent to participate in a ScheduleItem. The level is reflected in the Agent's Status. For example, if the ScheduleItem is a meeting, the values for this Status might be "invited", "confirmed", or "declined"

Method Summary

Id	getAgentId()
Type	getStatus()

org.osid.scheduling.Timespan

The Timespan returns the start and end of a ScheduleItem.

Method Summary

long	getEnd()
long	getStart()

org.osid.scheduling.*Object* Iterators

Iterators return a set of elements of a specific type, one at a time. The purpose of all Iterators is to offer a way for SID methods to return multiple values of a common type and not use an array. Returning an array may not be appropriate if the number of values returned is large or is fetched remotely. Iterators do not allow access to values by index, rather you must access values in sequence. Similarly, there is no way to go backwards through the sequence unless you place the values in a data structure, such as an array, that allows for access by index.

All iterators contain two methods. The **hasNext<*Object type*>**() method returns true if there are more values of the iterator type available; false otherwise. The **next<*Object type*>** () method returns the next element in the sequence. Note that in many cases, the order of elements is not guaranteed. The following iterators are included in this SID:

- AgentCommitmentIterator
- ScheduleItemIterator
- TimespanIterator

Type iterators, such as that used for Message and Delivery Types are defined in osid.shared.TypeIterator.

org.osid.scheduling.SchedulingException

The OSIDs make use of Exceptions as a mechanism for responding to error or unusual conditions. All methods in the Hierarchy OSID throw a **SchedulingException**. The Exception contains a message that is a String. The following message Strings are defined in SchedulingException:

Exception Message Summary

Constant	Message String
END_BEFORE_START	End cannot precede Start

If an implementation uses these messages, consumers of the implementation can easily test and conditionally respond to the Exception. Note that other kinds of Exception constructors are not used as all do or can devolve to a String. All methods of all interfaces of all OSIDs throw a subclass of org.osid.OsidException. This requires the caller of any implementation method handle the Exception.

If a method performs an operation without incident, an object or primitive may be returned, but in most cases, methods do not return error codes or a success or failure boolean. For example, a method that deletes an object with a particular identifier, would throw an Exception if the identifier were unknown; the method would not return, for example, false.