OSID Specification Memo: 3
Track: Framework
Status: Final

Tom Coppeto
OnTapSolutions

Scott Thorne
Massachusetts Institute of
Technology

January 2006

Structure and Definitions of OSIDs (S-OSID)

Status

This document is a specification of the OSID framework.

OSID framework specifications are used to describe, constrain, and inform the design of OSIDs.

Abstract

This document defines a structure and syntax for OSID definitions, the S-OSID. Specifications concerning OSID interfaces are written in the language described by this specification. This document also contains an XML schema representation of the S-OSID.

Table of Contents

1. Introduction

OSID specifications define a service using one or more interfaces. This document specifies the structure and syntax of OSID definitions.

OSID specifications are written neutral to a particular programming language and are bound to a language using the L-OSID specification that defines the transformation rules. It is expected that for a language binding to be successful, the language needs to minimally support the following:

- a primitive type such as a byte or char.
- complex types that contain methods
- arrays of primitive and complex types
- method parameters that support primitive and complex types, or pointers
- method return values that support primitive and complex types, or pointers

Different languages may offer varying degrees of type checking and interface enforcement. It is outside the scope of this specification to address these variances except to specify the definitions so that the OSID definitions are transformable.

2. OSID Definition

An OSID corresponds to a service package and must specify one or more interface definitions as defined in an OSID specification. An OSID contains the following components:

2.1. namespace

The `name` identifies the service package and provides the means for the OSID and L-OSID specifications to identify interfaces and organize packages. The name must be unique across all OSIDs.

The name of an OSID should reflect a service domain. Acronyms and abbreviations should be avoided and one-word labels are strongly encouraged.

2.2. title

The `title` is the formal name of the OSID used primarily for documentation.

2.3. version

The `version` identifies the revision level of the OSID using a string in the format `n1.n2.n3` where `n1` is an integer representing the major version number, `n2` is an integer representing the minor version number, and `n3` is an integer representing a maintenance version.

2.4. status

The `status` indicates whether this is a new or deprecated OSID.

2.5. copyright

A copyright is text containing any copyright information associated with this OSID.

2.6. license

The license contains licensing information for the OSID definition.

2.7. description

The description offers an overview in descriptive text on the purpose, behavior, and usage of this OSID. The purpose of the description is for inline documentation and may contain rich text (see Descriptions).

2.8. interface(s)

An OSID contains one or more interface elements.

3. Interface Definition

An interface is a set of methods and contains the following elements.

3.1. name

The name identifies the interface name. The interface name is composed of the OSID package name plus an extra name component to identify the interface within an OSID package. Thus, the interface name is unique across all the OSIDs. It is desirable although not required for the interface name component to be unique across all OSIDs.

Acronyms and abbreviations are strongly discouraged.

3.2. implements

An interface may extend another interface. Implements identifies the interface by name whose methods should be included within this interface.

3.3. status

The status indicates whether this is a new or deprecated interface.

3.4. description

The description offers an overview in descriptive text on the purpose, behavior, and usage of this interface. The purpose of the description is for inline documentation and may contain rich text.

3.5. method(s)

An interface contains one or more method elements.

4. Method Definition

A method is an operation within an interface that is invoked by an OSID Consumer. A method contains the following components:

4.1. name

The name identifies the method call. It must be unique among all name elements within an interface.

4.2. description

The description offers an overview in descriptive text on the purpose, behavior, and usage of this method. The purpose of the description is for inline documentation.

4.3. parameter

A method has zero or more parameter elements. See Parameter Definition.

4.4. return

A method has zero or one return values. A return value must be an OSID interface or a primitive type (See Primitive Type Definition).

4.5. error

A method must return its specified value if one is defined. If a method operation cannot return its specified value, a method may result in an error. An error is an exceptional condition that occurs when the method cannot satisfy the interface contract. See Error Definition.

4.6. compliance

The compliance indicates whether or not this method must be implemented for this interface to be compliant with the OSID specification. Some methods may be optional in which case the OSID Consumer may not be able to rely on it across OSID Providers. See Compliance Definition.

4.7. status

The status indicates whether this is a new or deprecated method.

5. Parameter Definition

Parameters define the arguments supplied to methods.

5.1. description

The description summarizes the use of the parameter.

5.2. array

The array flag is a boolean value signifying if the parameter is a set. Valid array values are true and false.

5.3. type

The type must be a primitive type (see Primitive Type Definition) or an OSID interface.

6. Error Definition

Errors are the result of an exceptional condition that occurs when a method cannot satisfy its contract. The means for transmitting errors is determined by the OSID Language Binding Specification (L-OSID).

6.1. type

The type identifies an error condition. Errors are grouped into four categories:

1. **User Errors:** Errors pertaining to a user of an OSID Consumer that are the result of a user action.
2. **Operational Errors:** Errors pertaining to a user of an OSID Consumer that are the result of an operational failure in an OSID Provider that although is undesirable, is unavoidable.
3. **Consumer Contract Errors:** Programming errors in the operation of the OSIDs by an OSID Consumer resulting in noncompliance with the contract. Proper consumption and OSID Consumers should avoid these errors.
4. **Provider Contract Errors:** Programming errors in the OSID Provider resulting in noncompliance with the contract. Proper consumption and OSID Providers should avoid these errors.

The following table lists the allowable errors in OSID method signatures.

Error Category	Error	Description
User	ALREADY_EXISTS	Attempt to add an object that already exists.
User	NOT_FOUND	A lookup of an object or relation by its identifier(s) was not found.
User	PERMISSION_DENIED	An authorization failure occurred.
Operational	CONFIGURATION_ERROR	An error occurred in configuring an OSID Provider during initialization.
Operational	OPERATION_FAILED	A problem occurred in the execution of a method. These are unavoidable problems such as a server crash, a database offline, or a DNS failure.
Operational	TRANSACTION_FAILURE	An error occurred while committing a transaction.

Error Category	Error	Description
ConsumerContract	ILLEGAL_STATE	A method has been executed at an inappropriate time.
ConsumerContract	INVALID_ARGUMENT	A bad value supplied to an OSID method parameter.
ConsumerContract	INVALID_METHOD	An undefined method was invoked through the OSID boundary.
ConsumerContract	NO_ACCESS	An attempt at updating a read-only value.
ConsumerContract	NULL_ARGUMENT	A null argument was provided to an OSID method.
ConsumerContract	UNIMPLEMENTED	An OSID method was invoked but not implemented by an OSID Provider. These errors can only be defined when the compliance is optional.
ConsumerContract	UNSUPPORTED	An OSID Provider does not support the supplied data or data type.
ProviderContract	BAD_LOGIC	An OSID Provider exhibited incorrect behavior.
ProviderContract	INVALID_ERROR	An OSID Provider returned an error not defined in an OSID method.
ProviderContract	INVALID_RETURN	An OSID Provider returned in invalid return value.
ProviderContract	MISSING_METHOD	A defined method is missing from the OSID Provider.
ProviderContract	NULL_RETURN	The OSID Provider returned a null.

6.2. description

The description describes the condition that results in this error.

7. Compliance Definition

The compliance indicates if a method must be implemented for this interface to be compliant with the OSID specification. Valid values are:

7.1. mandatory

A mandatory method must be implemented such that it results in a valid return or one of the defined errors. Mandatory methods cannot define an UNIMPLEMENTED error.

7.2. optional

An optional method may not be available for execution. Optional methods may result in an UNIMPLEMENTED error if not implemented by an OSID Provider.

8. Status Definition

The status provides versioning information for OSIDs, interfaces, enumerations, and methods.

8.1. new

The OSID, interface, enumeration, or method was added since the last release.

8.2. deprecated

The OSID, interface, enumeration, or method is deprecated and should no longer be used.

9. Primitive Type Definition

Some method parameters and returns will require the use of basic primitive types to encapsulate basic data or to describe an OSID object. These types may or may not be found in all language bindings on all platforms in which case the type may not be directly mapped to a language primitive in the L-OSID. For example, string may map to java.lang.String in Java and char * in C.

It is the job of the L-OSID specification to map these types into a corresponding primitive or complex object to deliver the minimal range requirements defined in this specification. The OSID Provider translates the values as found in a backend system into the appropriate types and objects defined in the L-OSID for transport through the interface. Some languages do not utilize typing in which case the L-OSID may map all of these primitives to the single native object (such as a string), for example.

9.1. boolean

A boolean is a truth value of true or false.

9.2. byte

A byte is a basic unit of storage supporting a minimum of an 8-bit value. A byte should be used to represent a unit of arbitrary data not defined in the OSIDs.

9.3. cardinal

A cardinal is 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.

9.4. decimal

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).

9.5. integer

An integer is a number supporting a 64-bit value (-9,223,372,036,854,775,808.. 9,223,372,036,854,775,808).

9.6. object

An object is an undefined value used in circumstances where it is not possible to specify an interface as a parameter or return. Use of object is discouraged in the design of the OSIDs.

9.7. string

A string is a sequence of zero or more display characters. Each display character should support an international character set.

9.8. timestamp

A timestamp is a date and time with millisecond precision supporting the Quaternary period of the Cenozoic era (2,000,000B.C..2,000,000A.D). When this period will end is a simple approximation and perhaps wishful thinking. OSID Providers, however, may select to narrow the scope to the Holocene epoch of this era (10,000B.C).

OSID Providers should use timestamp when expressing an actual recorded date and time. Year representations where the day is irrelevant should be expressed using integer. A time without a corresponding date is a cardinal measuring units of time from midnight. A date without a corresponding time may be expressed as a timestamp with no time component.

10. Special Constructs

10.1. null

null values are not valid in any method parameter or return in any OSID specification.

10.2. array

OSID specifications may define an array of primitive or complex types, or other OSID objects, in a method parameter. The OSID specifications should assume that the length of the array could be determined from the array itself. In languages where this is not possible, the OSID-L should address this with an appropriate transformation.

10.3. enumeration

An enumeration may be used to indicate a restricted set of values in a method parameter or return where the enumeration object and its values are defined in and constrained to an OSID specification. The S-OSID treats an enumeration as an OSID defined object and the OSID specification refers to the enumeration using its name. An L-OSID may or may not be able to enforce the enumerated values in a language binding.

11. Descriptions

All textual descriptions may utilize a markup language to supply an L-OSID with formatting hints. An L-OSID may create inline code documentation or external documents based on the description. The following components lists the formatting tokens.

11.1. heading

A heading is used to label a documentation section.

11.2. code

Code indicates that the white space in the text must be preserved. An L-OSID may display code in a different font or enclosed in a text box.

11.3. token

A token indicates the word or phrase is a keyword and should be emphasized.

11.4. pbreak

A pbreak inserts a new paragraph marker.

11.5. link

A link is a hypertext link that may be used by an L-OSID binding to an HTML format. It contains the following components.

11.6. ref

The ref points to the referred hypertext document.

11.7. display

The display indicates what text to display instead of the ref contents. The display is optional and defaults to ref.

11.8. outline

An outline is a list of text elements to be displayed as a list.

11.9. element

An element of the outline.

In addition to the above elements, XML character entity references are also permitted.

12. Schema

XOSID Schema

```
namespace xosid = "urn:inet:osid.org:schemas/osid/3"

grammar {
    start = osid

    osid = element xosid:osid {
        attribute xosid:name { text },
        element xosid:title { text },
        attribute xosid:version { "3.0.0" },
        osidStatus?,
        element xosid:copyright { osidText },
        element xosid:license { osidText },
        element xosid:description { osidText },
        meta?,
        (osidInterface* & osidEnumeration*)
    }

    meta = element xosid:meta {
        osidPrimitives,
        osidErrors,
        osidStatements
    }
```

```
    meta = element xosid:meta {
        osidPrimitives,
        osidErrors,
        osidStatements
    }
```

```
    osidEnumeration = element xosid:enumeration {
        attribute xosid:name { text },
        element xosid:description { osidText },
        osidStatus?,
        osidEnumItem+
    }

    osidEnumItem = element xosid:item {
        attribute xosid:name { text },
        element xosid:description { osidText },
        osidStatus?
    }
```

XOSID Schema

```
osidInterface = element xosid:interface {
    attribute xosid:name { text },
    osidImplements*,
    osidStatus?,
    element xosid:description { osidText },
    osidMethod*
}

osidImplements = element xosid:implements {
    attribute xosid:interface { text }
}
```

```
osidMethod = element xosid:method {
    attribute xosid:name { text },
    element xosid:description { osidText },
    osidParameter*,
    osidReturn?,
    osidError*,
    osidCompliance,
    osidStatus?,
    element xosid:implNotes { osidText }?
}

osidParameter = element xosid:parameter {
    attribute xosid:name { text },
    (osidInterfaceType | osidPrimitiveType),
    element xosid:description { osidText }
}

osidReturn = element xosid:return {
    (osidInterfaceType | osidPrimitiveType),
    element xosid:description { osidText }
}

osidInterfaceType = element xosid:interfaceType {
    attribute xosid:type { text },
    attribute xosid:array { xsd:boolean }?
}
```

XOSID Schema
```
osidPrimitiveType = element xosid:primitiveType {
    attribute xosid:type { "boolean" |
                           "byte" |
                           "cardinal" |
                           "decimal" |
                           "integer" |
                           "object" |
                           "string" |
                           "timestamp"
    },
    attribute xosid:array { xsd:boolean }?
}
```
```
osidPrimitives = element xosid:primitives {
    element xosid:description { osidText},
    osidPrimitive*
}

osidPrimitive = element xosid:primitive {
    attribute xosid:type { text },
    element xosid:description { osidText }
}
```
```
osidErrors = element xosid:errors {
    element xosid:description { osidText },
    osidErrorCategory*,
    osidError*
}

osidErrorCategory = element xosid:errorCategory {
    attribute xosid:type { text },
    element xosid:description { osidText }
}
```
```
osidError = element xosid:error {
    ((
        attribute xosid:category { "User" },
        attribute xosid:type { "ALREADY_EXISTS" |
                               "NOT_FOUND" |
                               "PERMISSION_DENIED"
        }
    ) |
    (
        attribute xosid:category { "Operational" },
        attribute xosid:type { "CONFIGURATION_ERROR" |
                               "OPERATION_FAILED" |
                               "TRANSACTION_FAILURE"
        }
    ) |
```

XOSID Schema

```
            (
                attribute xosid:category { "ConsumerContract" },
                attribute xosid:type { "ILLEGAL_STATE" |
                                       "INVALID_ARGUMENT" |
                                       "INVALID_METHOD" |
                                       "NO_ACCESS" |
                                       "NULL_ARGUMENT" |
                                       "UNIMPLEMENTED" |
                                       "UNSUPPORTED"
                }
            ) |
            (
                attribute xosid:category { "ProviderContract" },
                attribute xosid:type { "BAD_LOGIC" |
                                       "INVALID_ERROR" |
                                       "INVALID_RETURN" |
                                       "MISSING_METHOD" |
                                       "NULL_RETURN"
                }
            )),
            element xosid:description { osidText }
        }
```

```
        osidStatements = element xosid:statements {
            element xosid:description { osidText },
            osidStatement*
        }

        osidStatement = element xosid:statement {
            attribute xosid:category { text },
            attribute xosid:type { text },
            element xosid:description { osidText }
        }

        osidCompliance = element xosid:compliance {
            attribute xosid:type { "mandatory" | "optional" },
            element xosid:description { osidText }
        }

        osidStatus = element xosid:status {
            attribute xosid:type { "new" | "deprecated"},
            element xosid:asOf { text },
            element xosid:description { osidText }
        }
```

XOSID Schema

```
        osidText = mixed {
            osidOutline* &
            osidLink* &
            element xosid:heading { text }* &
            element xosid:code { text }* &
            element xosid:token { text }* &
            element xosid:pbreak { empty }* &
            element xosid:copyrightSymbol { empty }*
        }

        osidOutline = element xosid:outline {
            element xosid:element { osidText }+
        }

        osidLink = element xosid:link {
            element xosid:ref { osidText },
            element xosid:display { osidText }
        }
    }
```

13. References

Shubert, Charles H., *XOSID 2.1.0,* May 21, 2005.

Coppeto, T., Thorne, S., *OSID Specification Framework*, October 2005.

14. Copyright Statement